Agentes de Transmisión de Correo

Trabajo de grado de la Licenciatura en Informática
de la Facultad de Ciencias Exactas
Universidad Nacional de La Plata

Alumno: Adrián Gustavo Russo

Director: Lic. Javier Díaz

Mayo de 1994

INDICE

Prefacio

Las redes de computadoras son unos de los pilares del desarrollo de la informática aplicada. Dentro de esta temática, un punto que cada año adquiere mayor relevancia son las redes académicas. El ejemplo mas claro del interés y potencialidad es Internet con más de 1.500.000 computadoras interconectadas con líneas dedicadas alrededor del mundo. La Universidad Nacional de La Plata no es ajena a los cambios ocurridos en los últimos años. A mediados de 1992 se dio inicio a lo que hoy denominamos red académica de la UNLP (a fines de 1992 nos encontrábamos con una red donde había solo cinco computadoras pertenecientes al CESPI y al Dto. de Física). Actualmente esta red se encuentra en pleno desarrollo y difusión (hoy tenemos en funcionamiento alrededor de 200 computadoras interconectadas, distribuidas en las facultades de Ingeniería, CESPI, Ciencias Exactas y Ciencias Astronómicas y Astrofísicas). Estas redes académicas ofrecen variados servicios a sus usuarios (investigadores, alumnos, laboratorios, etc), dentro de ellos se destaca el correo electrónico. Todas las redes existentes (BITNET, UUNet, Internet, HEPnet, etc) brindan este servicio. La atracción del correo electrónico es, obviamente, la posibilidad de un intercambio rápido de conocimiento entre grupos de investigación. Sin embargo, hay otras ventajas que no son tan conocidas como ésta. El teléfono, por ejemplo, también proporciona un acceso e intercambio inmediato de información, pero algunos estudios realizados han demostrado que aproximadamente el 75% de las llamadas fallan en el intento de alcanzar su propósito (" Lo siento mucho, pero el Señor Hugo Ramón no está en su laboratorio, se encuentra fuera de la ciudad en una reunión "). El correo electrónico tiene la misma velocidad que el sistema telefónico, pero a diferencia de éste, no necesita que los dos interlocutores se encuentren disponibles en el mismo instante.

Los sistemas de correo están compuestos básicamente por dos grandes subsistemas. El primero, son los User Agents (UA's). Estos tienen por objetivo tratar de abstraer al usuario de los grandes problemas de ruteos. También tiene por objetivo ofrecerle un ambiente de interacción y visualización para consultas de viejos mensajes, reenvios, manejos de alias de direcciones a nivel usuario, etc. El otro gran subsistema son los Agentes de Transmisión de Correo o Message Transfer Agents (MTA's). Estos son los encargados de interactuar con los UA's o con otros MTA's.

El objetivo de los MTA's es el de rutear los mensajes, realizar las transformaciones adecuadas hasta llegar a la dirección correcta y administrarlos. El último subsistema, el MTA, es considerado tabú para los administradores y es su existencia, desconocida para la gran mayoría de los usuarios.

El correo electrónico o como comúnmente se lo denomina *e-mail* (que proviene de las palabras del inglés electronic mail), debió adaptarse a los cambios y avances tecnológicos surgidos durante los últimos años. Originalmente los correos electrónicos estaban preparados solo para transmitir texto (como prueba de ello, nos encontramos con las especificaciones de diseño del Simple Mail Transfer Protocol, más conocido por sus siglas SMTP que sólo admite la transferencia de texto). El desarrollo tecnológico permitió a los usuarios una variada gama de servicios que estaban fuera del alcance del correo

electrónico (tales como fax, fotos digitalizadas, sonido digitalizado, etc), como así también el contar con un variado abanico de protocolos de correo electrónico (tales como X400.84 y X400.88 sobre ISO/OSI, SMTP sobre Transmision Control Protocol/Internet Protocol (TCP/IP), Unix to Unix Copy Program (UUCP),...). Se exigieron niveles de seguridad de usuarios y control las conexiones de los sistemas de transferencia. Ante esta nueva realidad, los viejos MTA's (por ejemplo: Sendmail) tuvieron que adaptarse para satisfacer estas demandas. Al mismo tiempo los nuevos MTAs (por ejemplo: Postman Pat) han sido diseñados para soportar estas posibilidades. Sin embargo estas cuestiones no se hallan totalmente resueltas por la constante aparición y/o modificación de nuevos protocolos.

Tres graves problemas dificultan el desarrollo de los MTA's: en primer lugar los diferentes formatos de mails (Request For Coments 733 (RFC-733), RFC-822, X400, X400/MOTIS, MIME, FAX, VOZ, etc), luego el mapeo entre el RFC-822 y X400 en función del los RFC-987, RFC-1148 y el RFC-1148bis y por último la conectividad de los intermediarios (que se utilizan como mecanismos de transporte por la variada gama de protocolos de red que se ven involucrados en la transmisión de los mails). Estos dos últimos dificultan la puesta a punto de estos.

La amplitud del campo de trabajo en MTA's requiere un esfuerzo significativo para su comprensión y uso eficiente en una organización del estilo de la UNLP.

El informe esta dividido en tres partes. La primera, trata de dar una introducción conceptual de los elementos fundamentales que componen los sistemas de correos sin entrar en detalle de los protocolos que se utilizan, fundamentalmente en el de comunicaciones ya que este no es nuestro objetivo principal. La segunda parte tratará de interrelacionar los elementos que han de ser descriptos en el punto anterior con implementaciones y experiencias realizadas en la red de la UNLP y por último la tercera parte que resumirá la funcionalidad de cada uno de los MTA's con mis conclusiones.

Objetivos

Todo el proyecto se basa en profundizar el aspecto de los MTA's en un entorno tipo Internet. Por tal motivo se requiere el manejo de temas vinculados al TCP/IP y Domain Names Servers (DNS). Como así también realizar tareas adicionales como la creación del subdominio física dentro de la red de la UNLP y la instalación de Names Servers (NS) sobre dicho subdominio.

El propósito del presente trabajo de grado es realizar un estudio comparativo de diversos MTAs (Sendmail, Multichannel Memorandum Distribution Facility (MMDF) y Postman Pat (PP)) y analizar el grado de dificultad de sus administraciones contrastándolas con sus prestaciones. Estos han sido diseñados para correr sobre ambientes UNIX. El Sendmail y el MMDF son MTAs estándares. El Sendmail se encuentra en una computadora i486 con Linux (perteneciente al LANADI - Laboratorio Nacional de Difracción - Dto. de Física de la UNLP) y en una computadora HP-UX 9000/700 Apollo 710 RISC (perteneciente al PROFIMO - Programa de Física Molecular - Dto. de Física de la UNLP). El MMDF se encuentra en una computadora i486 con Santa Cruz

Operation (SCO), perteneciente al Centro de Cómputos del Dto. de Física). El PP es un MTA de avanzada cuyos fuentes fueron traídos de Australia por FTP (File Transfer Protocol) para tal fin, conjuntamente con el ISODE-7.0. La compilación y puesta en funcionamiento del PP y del ISODE-7.0, se hizo en una computadora de la red HP-UX 9000/700 Apollo 710 RISC. El PP es un MTA que nunca fue instalado en una HP-UX 9000/700 , con lo cual se hicieron las correcciones y/o modificaciones para su puesta en funcionamiento. Se analizarán las posibilidades de la red TCP/IP con SMTP.Asimismo se verá la relación e interacción de estos servicios con los DNS que deberán ser configurados para la interacción de los MTAs sobre TCP/IP y para una posible conexión on-line en Internet.

Este trabajo tiene también por objetivo la decisión de configurar y elegir (fruto de los estudios que se realizarán) al Sendmail o PP, para ser instalado en el PROFIMO sobre la HP-UX 9000/700 Apollo 710 RISC. Y por último se configurará el correo del Dto. de Física para trasladarlo de una computadora VAX (que es la computadora que actualmente esta ofreciendo el servicio) a una computadora i486 perteneciente al Dto. de Física con UNIX SCO sobre MMDF con TCP/IP.

El presente proyecto busca sentar las bases sólidas, para un trabajo sistemático, en experiencias concretas sobre la red de la UNLP que servirá de base para futuros desarrollos, en temas vinculados a X400 y X500, mail-multimedia (MIME), y todo aquello que este relacionado con ISODE y e-mails.

Capítulo I
Aspectos Conceptuales

Los mensajes

Introducción

Los mensajes existen desde que existió el habla. Desde tiempos remotos los mensajes orales servían como único medio de comunicación entre grupos de personas. Cuando nace la escritura como nueva forma de expresión, los mensajes sufrieron un proceso de adaptación. Luego, con el correr de los siglos, se fue perfeccionando, su formato, los medios de distribución, hasta terminar en lo que hoy llamamos carta. No obstante el tiempo transcurrido desde los comienzos hasta la actualidad, los elementos que componen la carta o los mensajes no han sufrido grandes variaciones en su evolución.

Componentes que integran un mensaje

El contenido y el destinatario son los elementos principales del mensaje. Estos se encuentran fuertemente ligados entre sí (sin contenido no tiene sentido el destinatario y sin destinatario no tienen sentido el contenido) sin variar en sus definiciones generales con el correr de los años. El destinatario es una persona, entidad o grupo de personas, a la cual va dirigido el contenido del mensaje. Sin embargo la forma de dirigirse a este se ha ido perfeccionando a formas más complejas ajustándose a cambios socio_culturales. Es así que para la distribución formal de las cartas se hace imprescindible que el destinatario no sólo sea una persona, sino que éste venga acompañado de información adicional como país, ciudad, calle,... . Un elemento que se ha incorporado en forma explícita es el remitente (para que el destinatario conozca su procedencia). Estos tres elementos (contenido, destinatario y remitente) forman lo que hoy denominamos carta.

Mensaje electrónico

La aparición de las computadoras permitió la incorporación de los mensajes como medio práctico de comunicación (fomentado por los avances tecnológicos, fundamentalmente en el área de las redes de computadoras). Estas han asimilado los elementos que componen las cartas (cuerpo, destinatario, remitente) y han creado elementos complementarios alentadas por la tecnología a su disposición (por ejemplo: incorporación de clave de encriptado y direcciones de copia, identificadores,...). El mensaje, es el objeto elemental de los sistemas de correos. Sin estos no tendría ningún sentido el desarrollo de los grandes sistemas de correos de la actualidad. Esta nueva forma de comunicación se lo llamo e-mail. Dicho servicio constituye el principal medio de comunicación e intercambio de información entre usuarios mayoritariamente humanos.

Una equivalencia casi natural entre las cartas y los mensajes de los correos electrónicos es mostrada en la figura que esta a continuación. Esta trata de enfrentar los elementos que componen una carta tradicional con una aproximación del formato de los mensajes del correo electrónico.

a) Correo postal		b) Correo electrónico
Sr. Adrián Gustavo Russo Calle 62 # 630 La Plata. CP 1900 Argentina	DESTINATARIO	Nombre: Adrián Gustavo Russo Calle: 62 # 630 Ciudad: La Plata País: Argentina Clave de encriptado:1020993837
Prof. Dr. Blas Rivero Calle 115 y 47 La Plata. CP 1900 Argentina Título: Nombramiento del CONICET	REMITENTE	De: Prof. Dr. Blas Rivero Calle: 115 y 47 Ciudad: La Plata País: Argentina Título: Nombramiento del CONICET
Le hago llegar esta carta a los efectos de informarle la documentación que requiere el CONICET para tratar su nombramiento como profesional de apoyo. 1 Título secundario. 2 Titulo Universitario.	CUERPO	Le hago llegar esta carta a los efectos de informarle la documentación que requiere el CONICET para tratar su nombramiento como profesional de apoyo. 1 Título secundario. 2 Titulo Universitario.

Orígenes del correo electrónico

En sus comienzos el correo electrónico simplemente consistió en transferencias de archivos entre computadoras. Estos archivos eran los mensajes, donde por convención la primera línea era el destinatario o receptor. A medida que transcurrió el tiempo las limitaciones de este esquema se fueron haciendo más evidentes. Algunas de esas limitaciones fueron:

i) Resultaba trabajoso el poder enviar un mensaje a un grupo de personas.

ii) El emisor no podía saber si el mensaje era recibido o no por el destinatario.

iii) Era muy dificultoso el poder redireccionar los mensajes por problemas de ausentismo.

iv) La interfase de escritura y manejo de los mensajes era bastante rígida y pobre.

v) No era posible incorporar en el cuerpo del mensaje una mezcla de texto, voz digitalizada, gráficas, facsímil,... .

vi) No existía un estándar universal para el tratamiento de las direcciones o destinatario.

vii) No existía una forma de almacenamiento de los mensajes.

Pero tal vez una de las limitaciones más severas que tenía este sistema de correo electrónico primitivo, era la no estandarización de su estructura interna. El no contar con una estructura interna preestablecida, no se podía pensar en confeccionar algún sistema que hiciera las cosas más sencillas para el usuario (este tenía que editar el mensaje con algún editor del sistema y luego efectuar manualmente la transferencia del mensaje). Esto da inicio a las especificaciones

distintas marcas (IBM, HP, Digital, Apple, SUN,...) que utilizan diversos sistemas operativos (UNIX, VMS, VM,...).

Transcurrido cierto tiempo se fue ganando experiencia con estos sistemas, se hicieron nuevas propuestas para llegar a tener sistemas de correo electrónico más poderosos.

Interfases de visualización o Users Agents (UA's)

Introducción

En los inicios del correo electrónico, no existía la posibilidad de contar con un sistema con el cual se pudiera editar, visualizar, enviar y reenviar mensajes. Se debía básicamente a que no había una formalización establecida para la confección de los mensajes. Con la aparición de la estandarización de las estructuras internas y las direcciones de los mensajes, comenzaron a surgir los primeros sistemas de visualización o UA's.

Ventajas del uso de UA's

Los UA's sirven como herramientas para una fácil administración y visualización de los mensajes recibidos. Le permiten trabajar a los usuarios con los contenidos de los mensajes (visualizarlos, editarlos, salvarlos en archivos para su posterior procesamiento,...). Clasificar su correspondencia histórica según determinados criterios fijados por el usuario (por emisor, por tema, por fecha de recepción,...). Permitir diferentes formas de listar los mensajes recibidos,....

También los UA's tienen por finalidad darle al usuario un cierto nivel de abstracción en lo que se refiere al control y manejo de como ha de llegar el mensaje al destino (el usuario se desentiende de los mecanismos necesarios para que los mensajes lleguen al destinatario final).

En los ambientes UNIX se destacan dos UA's de uso masivo y de dominio público, ellos son, el *mailx* y el *elm* . Existen otros UA's propietarios que corren bajo X-Windows muy potentes y de fácil uso, como ejemplo tenemos al Z-Mail.

Alias y listas de correo

Para usuarios que hacen uso cotidiano del correo electrónico, resulta bastante dificultoso el recordar las direcciones electrónicas, de personas con las cuales se envian mensajes. Es más fácil recordar el nombre o sobrenombre de personas que sus direcciones. Es muy común que una persona cambie de dirección perdurando a estos cambios, su nombre o apodo. En consecuencia los UA's incorporan el concepto de agenda, para el manejo ágil de direcciones, es decir, contar con la capacidad de dado un nombre o apodo asociarle a él la dirección correspondiente.

Este concepto se lo conoce con el nombre de *alias*. Cuando se desea mandar un mensaje haciendo uso de un alias, el UA consulta las alias del usuario permutando el alias por la dirección correspondiente, sin que el usuario intervenga en el procedimiento. Como consecuencia los UA's adoptaron estas características, incorporando y/o modificando módulos que permitiesen el manejo y control de los alias (actualizaciones, visualizaciones,...).

listas de correo involucran usuarios donde todos tienen intereses comunes a esa lista, el ámbito de control y manejo de ella esta fuera del alcance de los UA's. Esto es así porque los alias que son manejados por los UA's, en general administran alias propias del usuario (por ejemplo: si el usuario A tiene definido en sus alias el alias profimo, con direccionamiento múltiple, cuando se envíe un mensaje con dicho alias, el UA reemplazará el alias por las direcciones correspondientes, ahora si un usuario B quiere hacer uso de ese alias porque desea enviar un mensaje al grupo profimo, este no podrá hacerlo debido a que los alias de A sólo son vistos por A, por lo que el usuario B deberá definir su propio alias profimo). Esta restricción no permite la creación de listas de interés, básicamente debido a que si la lista no tiene un alcance global (vista por todos los usuarios) no se podría hacer referencia a ella. La solución a la creación y administración de las listas de interés se verá más adelante

Casillas de correo o mailbox

De la misma forma que en las casas particulares se utilizan buzones o sitios determinados (casillas de correos), para que el cartero deje la correspondencia en ausencia del destinatario. En las computadoras se implementaron las casillas de correo (mailbox o folders) con el mismo criterio (el almacenamiento de mensajes). Cada usuario tiene asociado un mailbox . Su dirección determina el mailbox donde se han de recibir los mensajes. Existe una asimilación entre la dirección de usuario y su mailbox. Esto es así, porque el emisor busca como finalidad almacenar el mensaje en el mailbox del destinatario. Su uso permitió reemplazar la comunicación directa (interactiva) por el uso de transacciones que almacenan mensajes en el mailbox del destinatario. El sistema, al conectarse el usuario, le avisará de la existencia de correspondencia pendiente, y él podrá consultar su mailbox haciendo uso de los UA's que estén a su alcance.

La capacidad del uso del mailbox como mecanismo de almacenamiento de mensajes permitió que los usuarios sigan manteniendo su correspondencia aún

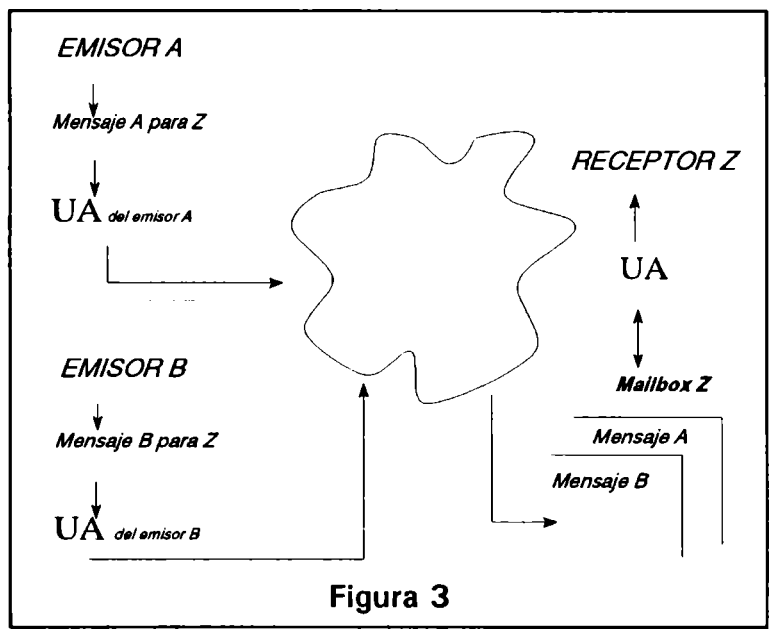

Figura 3

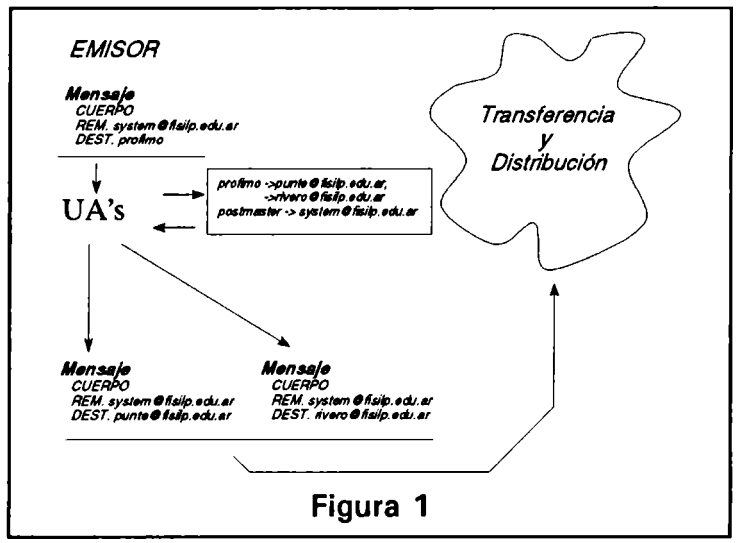

Figura 1

El concepto de alias fue ampliamente aceptado, y fueron incorporándose otras características. Una de ellas es la posibilidad de hacer referencia a múltiples direcciones. Cuando se envía un mensaje usando como dirección un alias múltiple, el UA sustituirá a este por las direcciones correspondientes.

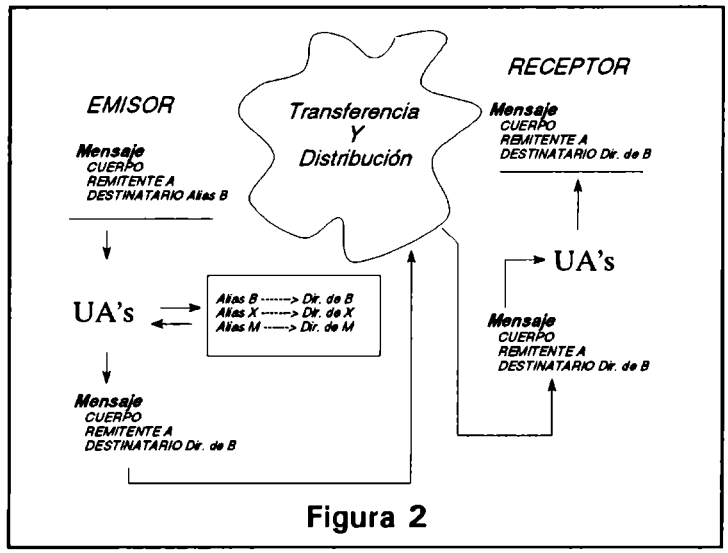

Figura 2

Por ejemplo: si se enviara un mensaje con dirección *profimo*, este indirectamente se enviará a las direcciones *punte@fisilp.edu.ar*, y a *rivero@fisilp.edu.ar*) Esta característica permitió que se incorpore el concepto de listas de correo. Las listas de correo fueron creadas con la finalidad de agrupar personas (accedidas a través de sus direcciones) que tienen características en común (integrantes de un laboratorio o de grupos de investigación,...). Cuando las

a su dirección, como hemos mencionado en párrafos anteriores, no obstante, los usuarios utilizan esta capacidad de almacenamiento como mecanismo de clasificación de su correspondencia almacenando los mensajes en mailbox's auxiliares definidos por el usuario según el criterio que este determine (por remitente, por tema,...), y administrados por los UA's.

Agentes de transmisión de correo o Messages Transfers Agents (MTA's)

Introducción

Los sistemas de visualización deben interactuar con otro sistema llamado agente de transmisión de correo o Message Transfer Agent (MTA) quienes son los que efectivamente resuelven como hacer para que un mensaje llegue a su correspondiente destinatario. Estos son los encargados de elegir los mecanismos adecuados para poder llegar a buen destino. También controlan el buen funcionamiento de los transportes que se han de usar como medio de comunicación. Administran y controlan el flujo global de los mensajes, ubicándolos en los mailboxs correspondientes o redireccionando mensajes a otros MTA's seleccionando el transporte que han de utilizar. Otra de sus funciones es la de controlar y chequear los formatos de las direcciones. La seguridad es también un tema muy importante en los sistemas de correos actuales. Es en los MTA's donde se controla y supervisa el buen funcionamiento y uso de los recursos disponibles para que los mensajes lleguen a buen destino.

Mecanismo básico de ruteo

Cuando dos computadoras se conectan en forma directa e intercambian mensajes, los problemas de direccionamiento para alcanzar el mailbox del usuario destino no es tan complicado. Los graves incidentes ocurren cuando son varias las computadoras (una red de computadoras interconectadas por diferentes medios) que intervienen para alcanzar el destino del mensaje. Fuertemente agrabado por los diferentes formatos de direcciones estándares que han aparecido con el correr de los años (X400(84), X400(88), RFC-822...), ya que estas son el elemento básico para que los mensajes lleguen a buen destino. A este problema se lo conoce con el nombre de ruteo. Para comprender este término presentaremos una analogía, por ejemplo: en el caso de una red telefónica, un nombre es el nombre de una persona, como Paula Quiroga, donde su número telefónico es visto como su dirección, y una ruta es una secuencia de líneas telefónicas, y conmutaciones que son usadas para llegar hasta el aparato de Paula Quiroga. En el caso de una red de computadoras un nombre corresponde a una persona o entidad, una dirección a su dirección electrónica, compuesta por una identificación dentro de su computadora y el nombre de la computadora, como mínimo, y la ruta es el conjunto de computadoras o nodos por los cuales debe pasar el mensaje para llegar a destino. Sin embargo, la mayoría de los usuarios no tienen conocimiento de cual será el recorrido de los mensajes, para llegar al destino. El direccionamiento, en general, es la búsqueda de una serie de MTA's a través de los cuales podamos alcanzar el mailbox del destinatario. Para un MTA en particular, esta tarea de direccionamiento consiste en encontrar otro MTA más próximo al que reenviar el mensaje, salvo que el destinatario sea local.

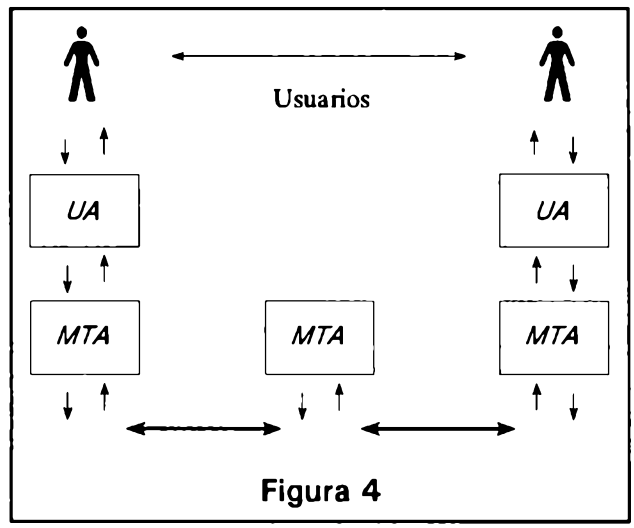

Figura 4

Los MTA's reconocen diferentes formatos estándares de direccionamiento, y es en los algoritmos de ruteo en donde los MTA's hacen las transformaciones de un formato estándar a otro. Por ejemplo: supongamos tener un usuario que posee dos tipos de direcciones una sobre un formato D1 y la otra sobre otro formato D2; sin embargo este usuario es reconocido en su computadora según su formato de dirección D1, con lo cual si arriba un mensaje con formato de dirección D2, el MTA deberá hacer, en las operaciones de ruteo, las transformaciones adecuadas para que el mensaje llegue al mailbox del usuario haciendo uso de D1. Existen reglas preestablecidas que realizan las transformaciones adecuadas de un formato de dirección a otro. Estas transformaciones serán explicadas más adelante cuando tratemos temas relacionados a los aspectos operacionales.

Es función del MTA como última actividad del algoritmo de ruteo depositar el mensaje en el mailbox del usuario.

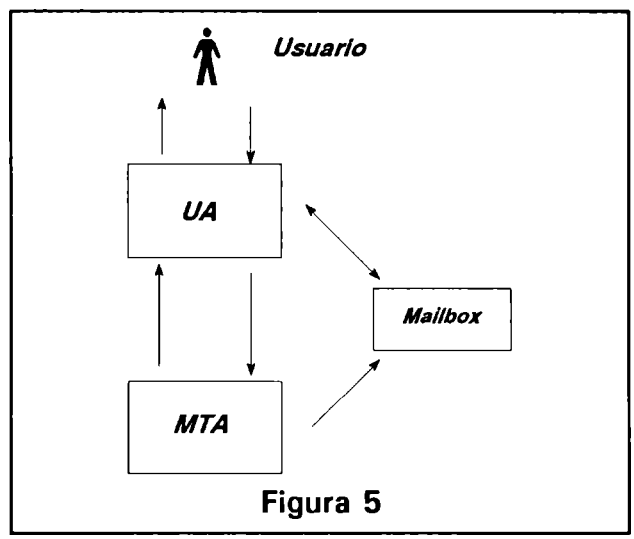

Figura 5

Alias y listas de correo del sistema

Los MTA's usan otros alias, que están fuera del alcance de los UA's, llamados alias del sistema. Los alias del sistema se encuentran por encima de los alias del usuario, asocian direcciones de interés común para todos los usuarios y para el administrador. Por ejemplo: el postmaster (administrador general del sistema de correo) podría generar un alias o lista de correo con todos los integrantes que hacen uso de este servicio dentro del sistema, para que cuando tenga que enviar un mensaje a todos, no tenga que enviarlo tantas veces como usuarios haya, sino una sola vez haciendo uso de él.

Hasta ahora no hemos profundizado en como se confeccionan las direcciones de correo electrónico, ya que no es la intención del capítulo I, pero para dar otro ejemplo de la funcionalidad de los alias del sistema vamos a hacer referencia a ellos sin entrar en detalle. Una dirección de correo electrónico podría ser opiro@ayelen.unlp.edu.ar donde, opiro es el usuario (Oscar Piro) de la máquina *ayelen* del PROFIMO. Sería recomendable, sobre todo en una red Internet, que los nombres de los usuarios no conformen la dirección electrónica, esto es así por un problema de seguridad y también por un problema nemotécnico. El alias puede ser usado en este caso para enmascarar la dirección *opiro@ayelen.unlp.edu.ar* como, por ejemplo, *Oscar.Piro@ayelen.unlp.edu.ar* donde esta dirección será vista por toda la comunidad. Cuando llega un mensaje a la dirección *Oscar.Piro@ayelen.unlp.edu.ar* el MTA hará el cambio necesario para que se haga efectivo el arribo del mensaje al mailbox correspondiente al usuario *opiro*

Existen MTA's, que incorporan el concepto de listas de correo como algo bien distinguido de los alias. En sus orígenes, el mantener una lista de correo no era algo muy complejo (contaba con muy pocos integrantes), pero con el correr del tiempo estas listas se fueron sobredimencionado (hoy se cuentan con listas de más de 2300 direcciones, por ej: pp-people@cs.ucl.ac.uk). La persona encargada de administrar el sistema de correo electrónico (postmaster) no cuenta con la capacidad física para poder actualizar permanentemente listas tan grandes de usuarios. En estos casos se le deroga la responsabilidad a un tercero quien se encarga de su administración. (altas, bajas y modificaciones). Con esto podemos apreciar que las listas de correo ya no son alias de referencia múltiples sino que también existen personas encargadas que la administran. Estos MTA's incorporan información adicional para enlazar los administradores con las listas que administran.

Seguridad y control

El concepto de seguridad es muy extenso y abarca una gran variedad de temas, desde el elemental control de acceso hasta la sofisticada auditoría informática. Los MTA's no escapan a ciertos controles de seguridad, ya que por lo general una vez configurados los mecanismos de ruteo, los MTA's trabajan en forma automática en las transferencias de los mensajes. Es ahí donde se debe ejercer mecanismos de autorización y control de acceso de los mensajes que han de ser transmitidos y/o retransmitidos a otros MTA's para que lleguen a destino. Por ejemplo: imaginemos tener un MTA A que para comunicarse con el mundo utiliza ciertos mecanismos de alto costo operacional (el teléfono), y este MTA se encuentra en una red de computadoras conviviendo con otros MTA's. Supongamos

mundo, si el MTA A no tuviera mecanismos de control y acceso, este no tendría manera de controlar que otros MTA's lo utilicen como puente, debiendo el MTA A afrontar todos los costos de transmisión. Es por eso que algunos de los MTA's que veremos en los próximos capítulos soportan esta característica.

Existen MTA's que poseen la capacidad de controlar el máximo volumen de información (mensajes) que puede pasar sobre todos o determinados medios de transporte (los medios de transporte se verán a continuación). Esto es así fundamentalmente por dos razones. La primera de ellas es que mensajes de volumen reducidos (aprox. 100Kb) son más manejables que mensajes de gran volumen (aprox. 1Mb) y en segundo lugar, como no se sabe a priori que medios se han de utilizar para que el mensaje llegue a destino, es recomendable transferir mensajes de tamaño reducido, pues si el MTA necesita reenviar el mensaje por problemas de interferencia es menos costoso reenviar un mensaje de volumen reducido que uno de gran volumen.

Medios de transporte

Introducción

Los MTA's tienen por objetivo principal el ruteo de los mensajes usando otros MTA's. Cómo se comunican? En párrafos anteriores hemos hecho referencia de que los sistemas de mensajería habían sido fomentados por los avances tecnológicos fundamentalmente en el área de las redes de computadoras. Durante las últimas tres décadas se han ido desarrollando y perfeccionando diferentes tipos de protocolos de comunicación. Estos son los que sirven como medio de transporte para la comunicación de los MTA's.

Uno de los protocolos de comunicación más popularmente usado fue el Unix to Unix Copy Program (UUCP). Tuvo una gran difusión a raíz del escaso equipamiento con el que se debía contar para ponerlo en funcionamiento y por su disponibilidad como software de dominio público. Si bien sus orígenes fueron los ambientes UNIX, ha sido adoptado como protocolo de comunicación por otros sistemas operativos tales como VMS (Digital), DOS (Microsoft),...(en el Dto. de Física de la UNLP se usa el UUCP sobre una Micro VAX Digital y en el CETAD se utiliza el UUCP sobre una máquina PC AT con DOS, el CESPI utiliza el UUCP sobre SCO).

Otro protocolo de comunicaciones que tuvo y tiene gran difusión es el Transmision Control Protocol / Internet Protocol (TCP/IP) es un protocolo de comunicación estándar de facto, con el cual se sustenta lo que hoy llamamos Internet. Los motivos de su aceptación casi inmediata fueron, su simpleza, su efectividad y fundamentalmente su aceptación en los ambientes UNIX como protocolo estándar de comunicaciones conjuntamente con el UUCP. También sirve (al ser aceptado por diferentes empresas de computadoras) como medio de comunicación estándar entre computadoras con diferentes sistemas operativos y diferentes arquitecturas. Como ejemplo de ello tenemos en la UNLP el gateway entre BITNET y UUNet con un enlace Slip sobre X.25 entre una IBM4381 con VM del CESPI y una i386 con Santa Cruz Operation (SCO) también del CESPI.

En 1977 la International Organization For Standardization (ISO) estableció un subcomité para desarrollar una arquitectura estándar en el ámbito de las comunicaciones cuyo resultado fue Open Systems Interconection (OSI). Este subcomité dio los conceptos básicos para la conectividad de sistemas abiertos para procesamientos de aplicaciones distribuidas llamado OSI Reference Model (OSI/RM). El OSI/RM es tomado como referencia para la comparación de protocolos y arquitecturas de redes. Se desarrolló el ISODE basándose en esas especificaciones. En nuestro caso daremos una introducción de la conformación de la arquitectura del modelo OSI/RM, luego desarrollaremos el modelo TCP-IP e ISODE.

El TCP/IP e ISODE son protocolos de comunicaciones de redes interactivas mientras que el UUCP es protocolo de comunicaciones no interactivo.

Conceptos de una arquitectura en capas

Las redes de computadoras están diseñadas en forma muy estructurada. La mayoría de las redes se organizan en una serie de capas o niveles, con objeto de

predecesora. El número de capas, el nombre, contenido y función de cada una varían de una red a otra. Sin embargo, en cualquier red, el propósito de cada capa es ofrecer ciertos servicios a las capas superiores.

La capa *n* en una máquina conversa con la capa *n* de otra máquina. Las reglas y convenciones utilizadas en esta conversación se conocen conjuntamente como *protocolo de la capa n*. A las entidades que forman las capas correspondientes en máquinas diferentes se las denomina *peer process*. En otras palabras, son los peer process los que se comunican haciendo uso del protocolo.

En realidad no existe una transferencia directa de datos de la capa *n* de una máquina a la capa *n* de la otra; sino, más bien, cada capa pasa la información de datos y control a la capa inmediatamente inferior, y así sucesivamente hasta que se alcanza la capa localizada en la parte mas baja de la estructura. Debajo de la capa n°1 esta el medio físico, a través del cual se realiza la comunicación real.

Entre cada par de capas adyacentes hay una interfase, la cual define los servicios y operaciones primitivas que la capa inferior ofrece a la superior. Cuando los diseñadores de redes deciden el número de capas por incluir en una red, así como lo que cada una de ellas deberá hacer, una de las consideraciones más importantes consiste en definir claramente las Interfases entre capas. Hacer esto, a su vez, requiere que cada capa efectúe un conjunto específico de funciones bien definidas. El diseño claro y limpio de una interfase, además de minimizar la cantidad de información que debe pasarse entre capas, hace más simple la sustitución de la realización de una capa por otra completamente diferente (por ejemplo: todas las líneas telefónicas se reemplazan por canales satelitales). Así todo lo que necesita de la nueva es que ofrezca exactamente el mismo conjunto de servicios a la capa superior contigua, tal y como lo hacía la antigua realización.

Al conjunto de capas y protocolos se los denomina arquitectura de red. Las especificaciones de esta deberán contener la información suficiente que le permita al diseñador escribir un programa o construir el hardware correspondiente a cada capa, y que siga en forma correcta el protocolo apropiado. Tanto los detalles de realización como las especificaciones de las Interfases, no forman parte de la arquitectura, porque se encuentran escondidas en el interior de la máquina y no son visibles desde el exterior. Más aún, no es necesario que las Interfases de todas las máquinas en una red sean iguales, suponiendo que cada una de las máquinas utilice correctamente todos los protocolos.

La idea de comunicación multicapa puede explicarse con facilidad mediante una analogía. Imagínese, por ejemplo, que hay dos filósofos, uno en Kenia y otro en Indonesia, que desean comunicarse (peer process de la capa n°3). Cada uno de ellos tiene que hacer uso de un traductor (peer process de la capa n°2), dado que no hablan el mismo lenguaje. Cada uno de ellos, a su vez, contacta a un ingeniero (peer process de la capa n°1). El filósofo A desea manifestar su afecto por los aryctolagus cuniclus a su igual. Para realizar esto, pasa un mensaje a su traductor (en Swahili) a través de la interfase 2/3, quien, dependiendo de protocolo de la capa n°2, podría interpretarlo como "Me gustan los conejos", "J'aime des lapins" o "Ik hou van konijngen".

El traductor pasa después el mensaje a su ingeniero para que lo transmita por telegrama, teléfono, red de ordenadores, o algún otro medio, dependiendo de lo que previamente haya acordado, con el otro ingeniero (protocolo de la capa

lo que previamente haya acordado, con el otro ingeniero (protocolo de la capa n°1). Una vez que llega el mensaje, se traduce al indonesio y se pasa a través de la interfase 2/3 al filósofo B. Debe apreciarse que cada protocolo es independiente de los demás, mientras no se cambien las interfases. Los traductores pueden conmutar del francés al holandés suponiendo que ambos estén de acuerdo y que ninguno modifique la interfase de la capa n°1 y la n°3.

Considérese ahora un ejemplo más técnico: cómo proporcionar comunicación a la capa superior de una red de siete capas.

Un proceso que se está ejecutando en la capa n°7 produce un mensaje *M*, el cual pasa de la capa n°7 a la capa n°6 de acuerdo con la definición de la interfase de la capa 6/7. La capa 6, en este ejemplo, transforma de cierta manera el mensaje (por ejemplo, mediante una compresión de texto), y lo pasa como nuevo mensaje *M* a la capa n°5, a través de la interfase 5/6. En este ejemplo, la capa n°5 no modifica el mensaje, sino que solamente regula la dirección de flujo (es decir, evita que algún mensaje de entrada sea considerado por la capa n°6, mientras esta se encuentra ocupada enviando una serie de mensajes de salida a la capa n°5).

En muchas redes no existe ningún límite en el tamaño de los mensajes que son aceptados por la capa n°4, sino más bien éste es impuesto por la capa n°3. Por consiguiente, la capa n°4 deberá dividir el mensaje de entrada en unidades más pequeñas y colocar un heder o cabecera en cada una de ellas. Esta cabecera incluye información de control cómo números de secuencia, mediante los cuales se logra que la capa n°4, en la máquina destinataria, pueda construir el mensaje mediante la colocación correcta de las unidades, si es que las otras capas no mantienen la secuencia. También en muchas capas, los headers o cabeceras contienen campos relacionados con el tamaño, tiempo y otros tipos de control.

La capa n°3 se encarga de decidir cuál de las líneas de salida va a utilizarse, le coloca sus cabeceras apropiadas y pasa los datos a la capa n°2. En la capa n°2 , no solo se añade un header a cada una de las unidades, sino que también una etiqueta al final y le entrega la unidad resultante a la capa n°1 para su transmisión física. En la máquina receptora el mensaje se mueve de capa en capa hacia la parte superior, y los headers se van retirando a medida que ascienden. Ninguno de los headers correspondientes a las capas inferiores de la n pasan a esta. La abstracción de los peer process es vital para el diseño de redes, sin ésta técnica sería difícil, si no es imposible, dividir el diseño de una red completa. Es decir, sería un problema intratable.

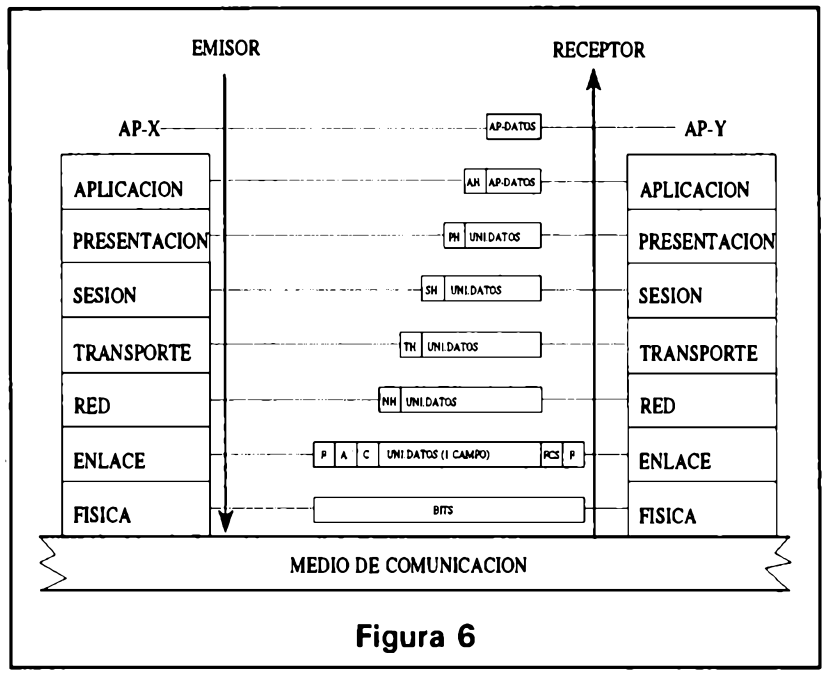

Figura 6

Modelo de referencia OSI

Una vez discutido las redes estructuradas en capas, de manera abstracta consideremos el modelo basado en una propuesta desarrollada por la International Organization for Standardization (ISO). A este modelo se lo conoce como Modelo de Referencia OSI de la ISO.

El modelo OSI/RM tienen siete capas. Los principios aplicados para el establecimiento de siete capas fueron:

i) Una capa se creará en situaciones en donde se necesita un nivel diferente de abstracción.

ii) Cada capa deberá efectuar una función bien definida.

iii) La función de las capas deberán seleccionarse tomando en cuenta la minimización del flujo de información a través de las interfases.

iv) El número de capas deberá ser lo suficientemente grande para que funciones diferentes no tengan que ponerse juntas en la misma capa y por otra parte deberá ser lo suficientemente pequeño para que su arquitectura no llegue a ser difícil de manejar.

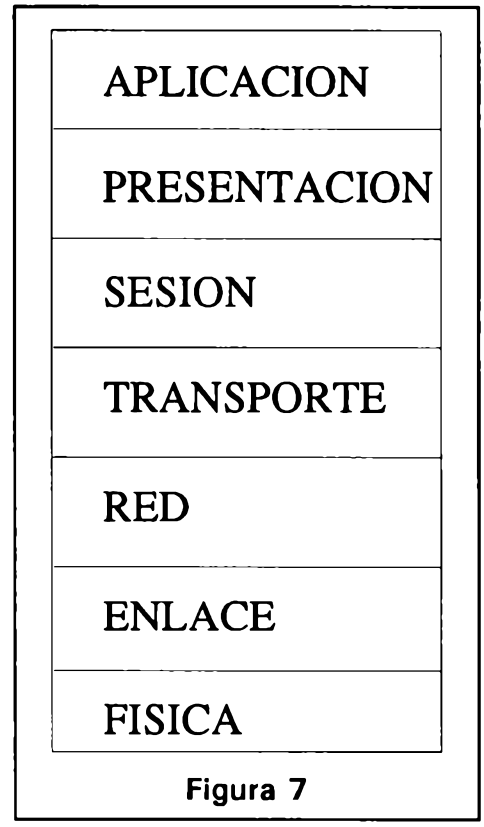

Figura 7

Capa física

La capa física se ocupa de la transmisión de bits a lo largo de un canal de comunicación. Su diseño debe asegurar que cuando un extremo envía un bit con valor 1, este reciba exactamente como un bit con ese valor en el otro extremo, y no como un bit de valor 0. Preguntas aquí son cuantos voltios deberán utilizarse para representar un bit con valor 1 o 0; cuantos microsegundos deberá usar un bit; la posibilidad de realizar transmisiones bidireccionales en forma simultanea,... .

Capa de enlace

La tarea primordial de la capa de enlace consiste en, a partir de un medio de transmisión común y corriente, transformarlo en una línea sin errores de transmisión para la capa de red. Esta tarea la realiza al hacer que el emisor divida la entrada de datos en tramas de datos y las transmita en forma secuencial y procese las tramas de confirmación, devueltas por el receptor. La capa física acepta un patrón de bits sin importarle su significado, es misión de la capa de enlace la de fijar los comienzos y límites de la trama mediante patrones de bits al comienzo y fin. Es también la encargada de controlar las retransmisiones y duplicaciones de tramas. Debe nivelar las diferentes velocidades del receptor y emisor.

Capa de red

La capa de red se ocupa del control de la operación de la subred. Un punto de suma importancia en su diseño, es la determinación sobre como encaminar los paquetes del origen al destino.

Capa de transporte

La función principal de la capa de transporte consiste en aceptar los datos de la capa de sesión, dividirlos, siempre que sea necesario, en unidades más pequeñas, pasarlos a la capa de red y asegurar que todos ellos lleguen correctamente al otro extremo. Además, todo este trabajo se debe hacer de manera eficiente, de tal forma que aísle la capa de sesión de los cambios inevitables a los que está sujeta la tecnología del hardware.

Capa de sesión

La capa de sesión permite que los usuarios de diferentes máquinas puedan establecer sesiones entre ellas. A través de una sesión se puede llevar a cabo un transporte de datos ordinarios, tal como lo hace la capa de transporte, pero mejorando los servicios que ésta proporciona y que se utilizan en algunas aplicaciones. Una sesión podría permitir al usuario acceder a un sistema de tiempo compartido a distancia o, transferir un archivo entre dos máquinas.

Capa de presentación

La capa de presentación realiza ciertas funciones que se necesitan bastante a menudo como para buscar una solución general para ellas, más que dejar que cada uno de los usuarios resuelva los problemas. Esta incluye funciones que son usadas por los programas de la capa de aplicación cuando hacen uso de la red. Se encarga de los aspectos de sintaxis y semántica. La ISO definió en esta capa el Abstract Sintax Notation (ASN.1) que provee una representación de datos para

que sea usado por la capa de aplicación. El ASN.1 es usado para salvar la diferencias de máquinas en la representación de los datos.

Capa de aplicación

La capa de aplicación contiene una variedad de protocolos que se necesitan frecuentemente. Por ejemplo: hay centenares de tipos de terminales incompatibles en el mundo. Otra función de la capa de aplicación es la transferencia de archivos. Distintos sistemas de archivo tienen diferentes formas de hacer referencia a el, así como diferentes formas de representar las líneas de texto,... . Este trabajo así como el correo electrónico corresponden a la capa de aplicación. En otras palabras los MTA's interactuan con la capa de aplicación de los transportes de redes.

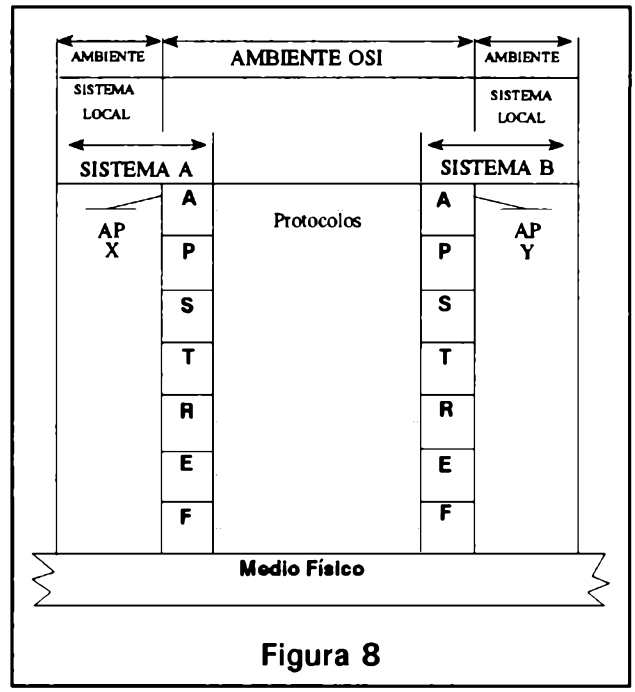

Figura 8

Arquitectura de los protocolos a usarse

Transmision Control Protocol/Internet Protocol (TCP/IP)

La arquitectura de capas del TCP/IP Internet defiere fundamentalmente de la arquitectura de capas de ISO en el número de capas que han sido definidas y en como fueron agrupadas las funciones que cada una de ellas realizan.

Existen cuatro capas, la capa de Aplicación, la capa de Transporte, la capa Internet, y por último la capa de Interface con la Network.

Capa de Aplicación

La capa de aplicación invoca a programas que acceden a servicios que son ofrecidos por TCP/IP Internet. Una aplicación interactúa con el protocolo de transporte para enviar y recibir datos. Cada programa de aplicación elige el estilo

de trasporte que necesita, pudiendo ser una secuencia individual de mensajes o un conjunto continuo de bytes (streams de bytes). El programa de aplicaciones pasa los datos a la capa de transporte quien se encarga de su entrega al destino.

Capa de transporte

La principal función de la capa de transporte es la de proveer que dos aplicaciones, una en un extremo y la otra en el otro, se comuniquen. Esta comunicación es llamada comunicación punto a punto. La capa de transporte puede regular el flujo de información. Este provee también transporte confiable asegurándose que los datos arriben al destino sin ningún error y en secuencia. La capa de transporte puede aceptar datos de muchos programas de aplicaciones y enviarlos a la capa de Internet en el mismo instante de tiempo.

La capa de internet

La función de la capa de internet es la de manejar la comunicación de una máquina a otra. Esta acepta pedidos de envío de la capa de transporte con la identificación de la máquina que se quiere alcanzar. Se encapsula el paquete con el header de la capa de internet llamado Internet Protocol (IP) y usa los algoritmos de ruteo para determinar si envía el paquete en forma directa o utiliza alguna máquina intermediaria para alcanzar el destino.

La capa de interface de red

Es la capa más baja del modelo de TCP/IP Internet. Recibe los paquetes IP y los transmite sobre una red específica. Esta capa puede consistir en un driver específico a la topología y a la placa usada.

Los MTA's interactuan con la capa de aplicación del TCP/IP a través de un protocolo diseñado para tal fin llamado Simple Mail Transfer Protocol (SMTP). Aquí no explicaremos su modo de operar ya que todavía no hemos formalizado ciertos aspectos en lo que se refiere fundamentalmente a el direccionamiento en un ambiente de red como TCP/IP. Estos conceptos serán explicados formalmente en el capítulo II del informe en donde se tratan los aspectos operacionales.

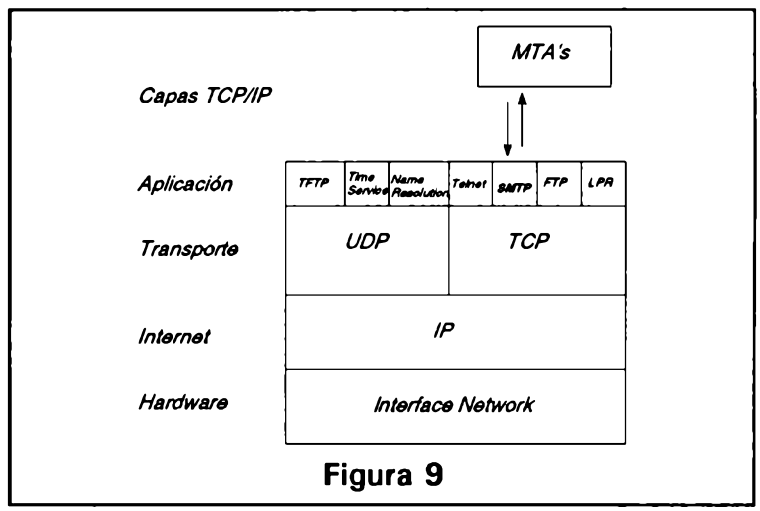

Figura 9

ISO Development Enviroment (ISODE)

Este paquete de software ha sido desarrollado con la intención de promover el uso y desarrollo de aplicaciones OSI; básicamente debido a la poca difusión que este tuvo.

La arquitectura en capas del ISODE se basa en hacer uso de protocolos de comunicaciones ampliamente aceptados por usuarios y empresas como los son TCP/IP Internet y X.25 hasta la capa de transporte y luego desarrolla las capas de sesión, presentación y aplicación (las funciones básicas de estas capas fueron ya descriptas en el modelo de referencia OSI). El ISODE para poder trabajar con funciones las cuales permitieran abstraerse de los mecanismos de transporte (TCP/IP Internet o X.25), definió una capa de software llamado TP0. Esta ofrece servicios a la capa de sesión quien para acceder al transporte lo hacen a través de ellas.

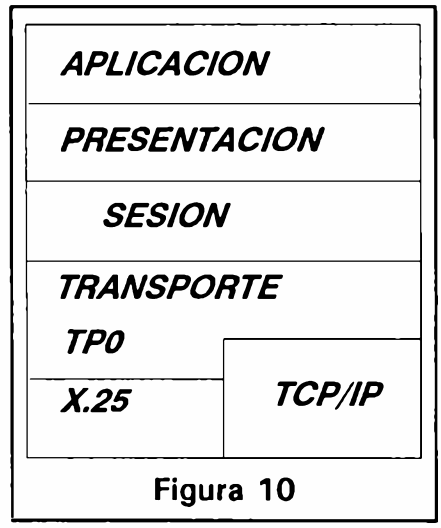

Figura 10

En este trabajo de grado tuvimos que traer el software del ISODE versión 7.0 y compilarlo sobre una plataforma RISC Apollo HP 9000/700 y crear las librerías tales como libtsap (para la interface entre TCP/IP y la de sesión, llamada TP4), la libssap (para la capa de sesión), la libpsap2 (para la capa de presentación).

UUCP

El UUCP más que un protocolo de comunicaciones es un sistema de comunicaciones de datos subordinado, muy potente y sofisticado que permite la transferencia de archivos entre computadoras. El UUCP es un paquete completo de movimiento de datos que puede transferir archivos de formatos ASCII y binarios entre máquinas y puede controlar la ejecución de comandos en la máquina remota. El hardware que usualmente se utiliza para hacer efectiva la comunicación son modems. No es interactivo con lo cual si se desea enviar un archivo a una computadora remota, haciendo uso de este protocolo, se deberá encolar dicha operación para cuando las máquinas se conecten se ejecuten las operaciones necesarias para satisfacer el requerimiento. De aquí se deduce que existe un cola de trabajo, que mantiene información de cuales son los archivos que intervienen en la transferencia y que operaciones asociadas poseen. El UUCP ofrece mecanismos ágiles de control, mantenimiento de la cola de trabajo para que cuando las máquinas se conecten se hagan la transferencias y operaciones asociadas. Existe una cola de trabajo por cada máquina remota. Su finalidad es la de organizar las operaciones y transferencias según la máquina con la cual me he de conectar o con la que ha de conectarse a mi.

El UUCP ofrece fuertes medidas de seguridad, herramientas completas de registros de conexión, de depuración, control de errores,... .

Capa de aplicación

La capa de aplicación son todos aquellos programas intervinientes para el control y administración de la cola de trabajo con el que cuenta el UUCP para ejecutar las transmisiones que le fueron encomendadas, o para que otra máquina transfiera información a través de este medio.

Capa de transporte

Es la capa donde se definen los transportes a utilizar para hacer efectiva la transferencia. En un principio el UUCP contaba con único transporte, el modem, hoy existen otros medios como lo son el TCP/IP, X25,... .

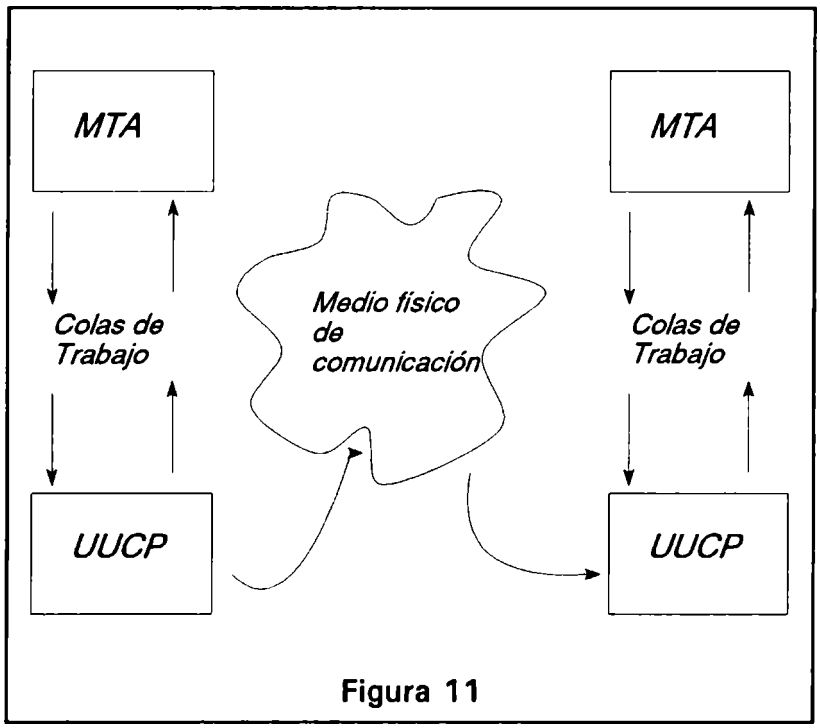

Figura 11

Capítulo II
Aspectos Formales y Operacionales

Mensajes

Introducción

En el capítulo I atribuimos que una de las limitaciones que tenían los usuarios de contar con un sistema de mensajería (dentro de los ambientes UNIX) que hiciera las cosas más sencillas era que los mensajes no contaban con una estructura interna preestablecida (estándar).

En los inicios de la década del 70, en los EEUU, fue la época de mayor crecimiento del servicio de correo electrónico. Cada empresa diseñaba sistemas de mensajería según su criterio, pero el Departamento de Defensa de los EEUU fue quien introdujo un estándar de facto, el RFC-822 (que explicaremos a continuación). Esta especificación fue adoptada por diferentes redes, como BITNET (Because It's Time Networks), UUNet (con UUCP), ARPA Internet (con TCP/IP).

En Europa al boom del correo electrónico se dio más tarde, a mediados de la década del 80, fundamentalmente debido al escaso equipamiento computacional y de comunicación con el que se contaba. Este retraso, permitió observar con atención los problemas de aplicación y funcionalidad del RFC-822. Como consecuencia la International Consultative Committee for Telegraphy and Telephony (CCITT) desarrolló otro estándar bajo el nombre de X400.

Actualmente el RFC-822 y el X400 son los estándares de especificación de estructuras internas de mensajes más importantes por su uso y difusión.

RFC-822

Convenciones Sintácticas y semánticas

A continuación daremos las definiciones de las reglas sintácticas que usaremos para formalizar las diferentes componentes de los mensajes en RFC-822.

Nombre de una regla

Los caracteres "<" y ">" no son usados en general. El nombre de la regla es simplemente el nombre en si mismo más que "<Nombre>". Los caracteres "<" y ">" son usados en las definiciones de la regla. La presencia de ellos nos facilitará discernir de una regla o no.

Regla1/Regla2, alternativas

Elementos separados por el carácter "/" son alternativos, por lo tanto "(Elemento1/Elemento2)" se leerá como Elemento1 o Elemento2.

(Regla1 Regla2), alternativos locales

Elementos encerrados con paréntesis son tratados como elementos simples. Entonces "(Elemento1 (Elemento2 /Elemento3) Elemento1)" se leerá como "Elemento1 Elemento2 Elemento1" o como "Elemento1 Elemento3 Elemento1".

*Regla, repetición

El carácter "*" precediendo un elemento indica repetición. La representación formal es "<1>*<m>Elemento". Significa que al menos 1 y no mas de m ocurrencias del elemento. El valor por omisión es 0 e infinito ("*Elemento).

"1*Elemento" significa desde 1 a infinito.

[Regla], opcional

Los corchetes encierran elementos opcionales, entonces [Elemento1 Elemento2] es equivalente a "*1(Elemento1 Elemento2)"

NRegla, repetición específica

"<n>(Elemento)" es equivalente a <n>*<n>(Elemento) o sea exactamente n (por ejemplo: 2DIGIT, son exactamente dos dígitos numéricos ASCII)

#Regla, listas

El carácter "#" es considerado constructor similar al "*" de la repetición cuya representación formal es "<1>#<m>Elemento", que indica que al menos 1 y a lo sumo m, donde cada uno de los elementos están separados por el carácter ",". Otra manera de representarlo con lo que ya hemos definido es "(Elemento *<m>("," Elemento))

";", comentarios

El carácter ";" a la derecha de la regla y hasta el final del renglón son comentarios.

Definiremos a continuación un conjunto de valores básicos (haciendo uso de las reglas sintácticas ya introducidas en los párrafos anteriores) con el cual trabajaremos en las definiciones sucesivas sobre el RFC-822. Estas, como así también las representaciones sintácticas las haremos sin traducirlas al español .

```
                                                    ; ( Octal, Decimal.)
CHAR        =  <cualquier carácter ASCII>           ; ( 0-177,  0.-127.)
ALPHA       =  <cualquier carácter alfabético ASCII> ; (101-132, 65.- 90.)
                                                    ; (141-172, 97.-122.)
DIGIT       =  <cualquier dígito decimal ASCII>     ; ( 60- 71, 48.- 57.)
CTL         =  <cualquier carácter de control ASCII y el DEL> ; ( 0- 37,  0.- 31.)
                                                    ; (   177,    127.)
CR          =  <ASCII CR, carriage return>          ; (    15,     13.)
LF          =  <ASCII LF, linefeed>                 ; (    12,     10.)
SPACE       =  <ASCII SP, space>                    ; (    40,     32.)
HTAB        =  <ASCII HT, horizontal-tab>           ; (    11,      9.)
<">         =  <ASCII quote mark>                   ; (    42,     34.)
CRLF        =  CR LF
LWSP-char   =  SPACE / HTAB
linear-white-space =  1*([CRLF] LWSP-char)
specials    =  "(" / ")" / "<" / ">" / "@"
               / "," / ";" / ":" / "\" / <">
               / "." / "[" / "]"
delimiters  =  specials / linear-white-space / comment
text        =  <cualquier CHAR, incluyendo CR y LF, pero NO CRLF >
atom        =  1*<cualquier CHAR sin los specials, SPACE y CTLs>
```

*quoted-string = <"> *(qtext/quoted-pair) <">*
qtext = <cualquier CHAR exeptuando <">, "\" y CR, y incluyendo linear-white-space>
*omain-literal = "[" *(dtext / quoted-pair) "]"*
dtext = <cualquier CHAR exeptuando"[", "]", "\" y CR, y incluyendo linear-white-space>
*comment = "(" *(ctext / quoted-pair / comment) ")"*
ctext = <cualquier CHAR exeptuando "(", ")", "\" y CR, y incluyendo linear-white-space>
quoted-pair= "\" CHAR
*phrase = 1*word*
word = atom / quoted-string

Descripción general

Internamente el mensaje esta compuesto por un encabezamiento (header) y opcionalmente por un cuerpo (body). El header y el body están separados por una línea nula (una línea con ningún carácter, precedida con los caracteres CRLF).

Cabeza o header

El header esta compuesto por campos. Estos pueden ser vistos como nombre del campo separados por el carácter ":", seguidos por el contenido del campo. El nombre del campo esta constituido por caracteres imprimibles ASCII (son los caracteres que van desde el 33 hasta el 126, exceptuando el carácter ":"). Ciertos contenidos de los campos pueden ser interpretados acorde a una sintaxis interna. Estos son llamados campos estructurados (por ejemplo: campos conteniendo la fecha o direcciones, que tiene un formato prestablecido). Otros campos, como el Subject y Comments, son simplemente secuencias de texto, a estos contenidos se los llama no estructurados.

La definición sintáctica de los campos como las de sus contenidos esta dada a continuación.

field	= field-name ":" [field-body] CRLF
field-name	= 1*<any CHAR, excluding CTLs, SPACE, and ":">
field-body	= field-body-contents [CRLF LWSP-char field-body]
field-body-contents	= <los caracteres ASCII hacen el field-body que consisten en combinaciones de atom, quoted-string y specials, o sino de texto>

A continuación representaremos sintáxticamente la composición del mensaje según este estándar.

*message = fields *(CRLF *text)*

fields = dates
source
*1*destination*

```
                    *optional-field
  source          = [ trace ]
                    originator
                    [ resent ]
  trace           = return
                    1*received
  return          =  "Return-path" ":" route-addr
  received        =  "Received"    ":"
                        ["from" domain]
                        ["by"   domain]
                        ["via"  atom]
                        *("with" atom)
                        ["id"   msg-id]
                        ["for"  addr-spec]
                        ";"    date-time
originator  =  authentic
               [ "Reply-To"  ":" 1#address] )
authentic   =  "From"      ":"  mailbox
               / ( "Sender"    ":"  mailbox
                   "From"      ":" 1#mailbox)

resent      =  resent-authentic
               [ "Resent-Reply-To"  ":" 1#address] )

resent-authentic =   "Resent-From"     ":"  mailbox
                   / ( "Resent-Sender"   ":"  mailbox
                     "Resent-From"     ":" 1#mailbox  )
dates       =  orig-date
               [ resent-date ]
orig-date   =  "Date"      ":"  date-time

resent-date =  "Resent-Date" ":"  date-time

destination =  "To"        ":" 1#address
               / "Resent-To"   ":" 1#address
               / "cc"          ":" 1#address
               / "Resent-cc"   ":" 1#address
               / "bcc"         ":"  #address
               / "Resent-bcc"  ":"  #address

optional-field =
               / "Message-ID"       ":"  msg-id
               / "Resent-Message-ID" ":"  msg-id
               / "In-Reply-To"      ":"  *(phrase / msg-id)
               / "References"       ":"  *(phrase / msg-id)
               / "Keywords"         ":"  #phrase
               / "Subject"          ":"  *text
               / "Comments"          ":"  *text
               / "Encrypted"        ":" 1#2word
               / extension-field
               / user-defined-field
```

```
msg-id      =  "<" addr-spec ">"

extension-field =  <Cualquir nombre que no comience con X->

user-defined-field =  <Cualquier nombre que no este definido aqui >


date-time   =  [ day "," ] date time       ; dd mm yy   hh:mm:ss zzz

day         =  "Mon"  / "Tue" /  "Wed"  / "Thu"
               / "Fri"  / "Sat" /  "Sun"

date        =  1*2DIGIT month 2DIGIT       ; por ejemplo: 20 Jun 82

month       =  "Jan"  /  "Feb" /  "Mar"  /  "Apr"
               / "May"  /  "Jun" /  "Jul"  /  "Aug"
               / "Sep"  /  "Oct" /  "Nov"  /  "Dec"

time        =  hour zone                   ; ANSI y Militar

hour        =  2DIGIT ":" 2DIGIT [":" 2DIGIT]  ; por ejemplo: 00:00:00 - 23:59:59

zone        =  "UT"  / "GMT"
               / "EST" / "EDT"
               / "CST" / "CDT"
               / "MST" / "MDT"
               / "PST" / "PDT"
               / 1ALPHA
               / ( ("+" / "-") 4DIGIT )

address     =  mailbox
               / group

group       =  phrase ":" [#mailbox] ";"

mailbox     =  addr-spec
               / phrase route-addr

route-addr  =  "<" [route] addr-spec ">"

route       =  1#("@" domain) ":"

addr-spec   =  local-part "@" domain

local-part  =  word *("." word)

domain      =  sub-domain *("." sub-domain)

sub-domain  =  domain-ref / domain-literal

domain-ref  =  atom
```

Observemos el siguiente ejemplo (nota: que cada uno de los campos que componen el encabezamiento ha sido resaltado. Dejamos al lector verificar que el encabezamiento del ejemplo coincida con la sintaxis anteriormente descripta).

Encabezamiento

From *atina.ar!uunet.uu.net!cs.ucl.ac.uk!pp-people-request@secyt*
Mon Jan 24 11:30:58 1994
Return-Path: *<atina.ar!uunet.uu.net!cs.ucl.ac.uk!pp-people-request@secyt>*
Received: from *unlp.unlp.edu.ar* ***by*** *ayelen.fisica.unlp.edu.ar* ***with*** *SMTP (PP)*
id *<00417-0@ayelen.fisica.unlp.edu.ar>;*
Mon, 24 Jan 1994 11:30:57 +0000
Received: from *secyt* ***by*** *unlp.unlp.edu.ar* ***id*** *aa04884; 24 Jan 94 11:24 ARG*
Received: from *atina ([140.191.2.2])* ***by*** *secyt.gov.ar* ***with*** *SMTP* ***id*** *<15137>;*
Mon, 24 Jan 1994 09:16:47 -0400
Received: from *uunet.uu.net* ***by*** *atina* ***with*** *UUCP (5.59/Vxxiii)* ***id*** *AA09404;*
Mon, 24 Jan 94 08:02:31 ARG
Received: from *haig.cs.ucl.ac.uk* ***by*** *relay1.UU.NET*
with *SMTP (5.61/UUNET-internet-primary)* ***id***
AAwafj14668;
Mon, 24 Jan 94 05:54:04 -0500
Received: from *bells.cs.ucl.ac.uk* ***by*** *haig.cs.ucl.ac.uk* ***with*** *local SMTP*
id *<g.00730-0@haig.cs.ucl.ac.uk>;*
Mon, 24 Jan 1994 09:00:12 +0000
Received: from *danpost4.uni-c.dk* ***by*** *bells.cs.ucl.ac.uk* ***with*** *Internet SMTP*
id *<g.24543-0@bells.cs.ucl.ac.uk>;*
Mon, 24 Jan 1994 08:59:58 +0000
Received: from *danpost4.uni-c.dk* ***by*** *danpost4.uni-c.dk* ***with*** *SMTP (PP)*
id *<00275-0@danpost4.uni-c.dk>;*
Mon, 24 Jan 1994 09:59:41 +0100
Date: *Mon, 24 Jan 1994 04:59:39 -0400*
From: *Erik Lawaetz <Erik.Lawaetz@uni-c.dk>*
Sender: *Erik Lawaetz <Erik.Lawaetz@uni-c.dk>*
Subject: *Re: Warnings template.....*
To: *Adam Bentley <ccx009@hermes.coventry.ac.uk>*
Cc: *pp-people@cs.ucl.ac.uk*
In-Reply-To: *<Pine.3.87.9401211754.A12569-0100000@rowan>*
Message-Id: *<Pine.3.89.9401240958.G29659-0100000@danpost4.uni-c.dk>*
Mime-Version: *1.0*
Content-Type: *TEXT/PLAIN; charset=US-ASCII*
Status: *RO*

(Linea de delimitación entre encabezamiento y cuerpo)

Cuerpo

> can someone tell me what I am supposed to put in a warnings
> template? I have a number of messages sitting on my hub from local users
> The following is defined in my tailor file for warnings...
>
> wrndfldir warnings
> chan warning prog=warnings,show="Send warning messages",
> type=warn

--Erik

Fin del mensaje

Funcionalidad de los campos

Redireccionamiento (fordwarding)

Algunos sistemas permiten a los receptores de mail redireccionar un mensaje (reteniendo los encabezamientos originales) mediante el agregado de un nuevo campo. Este estándar soporta este servicio a través del prefijo *Resent-*. *Resent_From* indica la persona que redirecciona (forwarded) el mensaje, mientras que el campo *From* indica el autor original.

Campos de seguimiento (trace fields)

Esta información es usada para proveer un seguimiento del manejo de mensaje. En consecuencia indica la ruta que el mensaje ha seguido desde su origen hasta su destino final.

Camino de retorno (Return-Path)

Este campo es agregado por el sistema de transporte que entrega el mensaje al mailbox del destinatario. Contiene la información definitiva acerca de la dirección y ruta de regreso del remitente. Podemos ver en el ejemplo anterior que el campo *Return-Path:* tiene como cuerpo la dirección de retorno *<atina.ar!uunet.uu.net!cs.ucl.ac.uk!pp-people-request@secyt>* .

Recibido (Received)

Una copia de esta campo es incorporada por cada MTA que reenvía el mensaje. La información de este campo puede ser de gran utilidad para detectar problemas de transportes (SMTP, UUCP,...). Los nombres de las máquinas de donde se envía y recibe el mensaje, conjuntamente con la fecha y hora, pueden ser especificadas con los parámetros *from*, *by*, ";". El parámetro *via* puede ser usado para indicar sobre que mecanismo físico el mensaje fue enviado (ARPANET, PHONENET) y el parámetro *with es* usado para indicar el protocolo de transporte utilizado en la conexión (UUCP, SMTP, X25). Algunos MTA's que encolan los mails (como lo veremos más adelante) le asignan al mensaje un número identificatorio (único en su propia máquina), este número o identificador a veces es incorporado a continuación del parámetro *id*. Analicemos un campo Received del ejemplo anterior. *Received: from unlp.unlp.edu.ar by ayelen.fisica.unlp.edu.ar with SMTP (PP) id <00417-0@ayelen.fisica.unlp.edu.ar>; Mon, 24 Jan 1994 11:30:57 +0000*, léase mensaje enviado de *unlp.unlp.edu.ar* a *ayelen.fisica.unlp.edu.ar* con protocolo de transporte *SMTP* con identificación interna de la máquina receptora *<00417-0@ayelen.fisica.unlp.edu.ar>* y con la fecha de recepción *Mon, 24 Jan 1994 11:30:57.*

Campos de emisión

Existen un conjunto limitado de combinaciones posibles con los campos *From, Sender, Reply-to, Resent-From, Resent-Sender* y *Resent-Reply-to*.

de (From / Resent-From)

Estos campos identifican la persona/s quien ha iniciado el mensaje. En el proceso de creación del mensaje, este campo se inicializará con la dirección de la máquina y el agente responsable de la emisión (persona, sistema o proceso). Sobre el ejemplo del mensaje que hemos dado tenemos *From: Erik Lawaetz <Erik.Lawaetz@uni-c.dk>* donde la dirección del emisor (el mailbox) es *Erik.Lawaetz@uni-c.dk* y donde además contamos con una aclaratoria, en este caso el nombre del emisor *Erik Lawaetz*.

Emisor (Sender / Resent-Sender)

Este campo contiene la identidad del agente (persona, sistema o proceso) que envia el mensaje. Comúnmente se lo utiliza cuando la persona que lo envia no es el autor del mensaje o para indicar quién, entre un grupo de personas, es el responsable del envío. La presencia del campo Sender podría ser completamente redundante con el campo From. El campo Sender debe estar presente si no lo está el campo From.

En la especificación del mailbox se incluye una secuencia de palabras que corresponden como identificación a un agente específico (por ejemplo: el nombre de un usuario o programa de computadora) más que una dirección estándar. En el campo Sender estas palabras identifican al agente, responsable del envío del mensaje con el agregado de la dirección de correo electrónico (siguiendo con el ejemplo dado tenemos: *Sender: Erik Lawaetz <Erik.Lawaetz@uni-c.dk>*).

Citemos dos ejemplos de la aplicabilidad de este campo: supongamos tener una secretaria que trabaja para la Licenciada Paula Quiroga cuyo nombre es Clara Oreda; la secretaria puede enviar un mensaje a nombre de Paula Quiroga y en el encabezamiento nos quedaría *From: Paula Quiroga <pquiroga@logic.com.ar>* (quien es el autor) y *Sender: Clara Oreda <coreda@logic.com.ar>* (quien es el responsable de la emisión). Otro ejemplo es cuando dos o más personas comparten la misma dirección de correo electrónico, la parte local de la dirección (local@domineo) puede ser no adecuado para referenciar a las persona que envian el mensaje, con lo cual las aclaraciones de las personas que la envian son de gran ayuda para identificarlas (por ejemplo: *Sender: L.Rizzi H.Ramon <lidi@unlp.edu.ar>*).

Reenviar a (Reply-To/Resent-Reply-To)

Este campo nos sirve como mecanismo general para indicar la dirección donde se quieren recibir las respuestas. Citaremos tres ejemplos de aplicabilidad de este campo. En el primero supongamos tener a una persona que no posee una dirección de correo electrónico estable, por lo tanto el puede mediante estos campos hacer referencia donde el quiere recibir la contestación de sus mensajes. En el segundo ejemplo el autor del mensaje puede hacer responsable de las respuestas de sus mensajes a otra persona (por ejemplo: en un congreso de Informática, el Presidente del congreso podría enviar correspondencia electrónica de temas organizacionales a sus colaboradores inmediatos y dar como dirección de recepción la de su secretaria). Como tercer y último ejemplo es cuando alguien esta subscripto a una lista de distribución entonces la dirección del campo Reply-To será la de la lista para asegurar una buena distribución de las respuestas a todos los integrantes de la lista.

Obsérvese la diferencia del campo Return-Path y la del Reply-To. El campo Return-Path es el camino del retorno del mensaje, generado al final del recorrido haciendo uso de los campos Received, mientras que el campo Reply-To es agregado sólo por la persona responsable de la emisión para indicar el lugar donde se quieren recibir las contestaciones.

Campos de recepción

Enviar a (To / Resent-To)

Este campo contiene la dirección de la persona a la cual le queremos enviar el mensaje.

Enviar una copia a (CC / Resent-CC)

Este campo contiene una dirección donde se quiere enviar una copia del mensaje.

Enviar copia adicional a (BCC / Resent-BCC)

Este campo contiene una dirección adicional donde se quiere que el mensaje llegue.

Campos de referencia

Identificación del mensaje (Message-Id / Resent-Message-Id)

Este campo contiene un identificador el cual hace referencia a la versión del mensaje. Esta identificación es única asegurada por la máquina que la genera.

En respuesta a (In-Reply-To)

El contenido de este campo identifica el mensaje que se ha contestado. Sobre el ejemplo tenemos *In-Reply-To: <Pine.3.87.9401211754.A12569-0100000@rowan>*, léase, en respuesta a tu mensaje con identificación *<Pine.3.87.9401211754.A12569-0100000@rowan>*te mando.....

Referencia a (References)

El contenido de esta campo nos indica otro identificador de mensaje al cual se hace referencia.

Claves (Keywords)

Este campo contiene frases, separadas por comas, para claves de identificación.

Otros campos

Título (Subject)

Este campo provee un resumen o indica la naturaleza del contenido del mensaje.

Comentarios (Comments)

Permite agregar comentarios sobre el mensaje sin alterar el contenido del cuerpo del mensaje.

Claves de encriptado (Encrypted)

Algunas veces es necesario ocultar los contenidos de los mensajes. Si el cuerpo del mensaje ha sido encriptado entonces en el encabezamiento del mensaje aparecerá el campo "Encripted" con una o dos palabras donde la primera indicará el software utilizado para el encriptado y la segunda si es que existe indicará la clave utilizada para el desencriptado (esta palabra de código será vista como una tabla indexada de claves que tendrá el receptor).

Direcciones

Introducción

En el capítulo I del informe dijimos que los mensajes estaban compuestos por destinatario, remitente y contenido, como así también que el destinatario es una dirección que nos permite llegar al destino, pero lo que hasta ahora no hemos mencionado es de que manera se confecciona esa dirección. Ahora hablaremos de como se fueron conformando cada uno de los formatos de las direcciones sobre diferentes medios de transportes y de los conflictos que existen sobre este tema.

Direcciones en TCP/IP (RFC-822)

En un ambiente de red sobre el protocolo TCP/IP (como lo es Internet) cada máquina en el mundo se encuentra univocamente identificada por un número de 32 bits, que por convención se escribe con cuatro números separados por puntos (por ejemplo: 140.191.52.160). Este identificador se lo conoce como *dirección de IP* (IP adress). Cuando un usuario desea conectarse con una máquina remota haciendo uso de este protocolo (TCP/IP), este lo debería hacer indicando su dirección de IP. Entonces cada paquete que circula a través de la red lo hace con los números de IP de la máquina origen y el de la máquina destino. Este esquema tuvo cierto rechazo de los usuarios, básicamente porque este sistema de identificación no era nemotécnico. En consecuencia, se diseño un mecanismo de identificación por nombre para las computadoras. En otras palabras, asociarle a cada máquina un nombre que también lo identifique univocamente en un ambiente Internet. Con ese nombre el usuario podría hacer referencia a la máquina sin que necesariamente conozca su dirección IP. Al conjunto de nombres de máquinas sobre Internet se lo denominó espacio de nombres, donde originalmente los nombres estaban compuestos por una secuencia de caracteres sin ninguna estructura en particular. El Network Information Center (NIC), administraba el espacio de nombres en forma centralizada, determinaba cuando un nombre era apropiado, que no fuera obsceno, que sea nemotécnico,... . La principal ventaja de este esquema era que los nombres eran apropiados y cortos. Pero existían motivos más que suficientes para cambiar rápidamente este sistema por razones administrativas y técnicas. En primer lugar el NIC era el único organismo autorizado para incorporar nuevos nombres, con lo cual la carga administrativa de mantenimiento aumento a medida que se iba incrementando el número de sitios que se incorporaban a la red. Para entender la severidad del problema, imagine un rápido crecimiento con miles de lugares, donde en cada lugar existen cientos de computadoras personales y workstations (estaciones de trabajo). En todo momento se conectan nuevas computadoras a red, cuyos nombres debían ser aprobados por la autoridad central (NIC). Otro motivo era el costo de mantenimiento por los frecuentes cambios y/o

modificaciones que se debía hacer a la base de datos que contenía los enlaces de los nombres de las máquinas con su número de IP. Si la base de datos se encontraba en un sólo lugar, entonces el tráfico de información se constituía en un cuello de botella cuando se incrementaba el número de lugares que se incorporaban a Internet con el agregado de la dificultad de su mantenimiento.

La pregunta que nos podemos hacer es como se diseña un sistema de nombres que se acomode a un conjunto grande de computadoras, que pueda acompañar el crecimiento o expansión exponencial de la red y que no requiera una administración central?. La respuesta es obvia, delegar autoridad para un espacio de nombres y distribuir la responsabilidad para el mapeo entre nombres y direcciones. Este sistema de distribución del espacio de nombres es un sistema jerárquico de nombres, el cual nos garantiza efectividad en el mapeo y nos asegura autonomía. Por ejemplo: para entender por que se planteo un sistema jerárquico pensemos en una estructura interna de una gran empresa (YPF, SEVEL, IBM,...). A la cabeza de la empresa esta un ejecutivo o presidente de la empresa, quien posee toda la responsabilidad, pero este no puede ver todo lo que sucede dentro de ella, con lo que se desprende que existen divisiones. Estas divisiones tienen un encargado o gerente de división donde su radio de acción es menor que el del presidente de la empresa. El presidente le asegura al gerente de división, autonomía dentro de determinados límites. Las divisiones pueden estar subdivididas en oficinas donde en cada oficina existe un encargado de sección con idénticas características que el gerente de división sólo que su alcance es inferior al del gerente. El gerente le delega responsabilidades y autoridad a los jefes de sección sin que dichas responsabilidades sean directamente dadas por el presidente de la empresa (ejercicio de la autonomía del gerente).

En Internet se particionó el espacio de nombres basados en sitios físicos y/o por actividad, y se le delegó a cada partición la responsabilidad y autoridad para el mantenimiento de nombres dentro de esa partición. Al delegar autoridad a cada división o partición, la raíz no se verá afectada por los cambios que ocurriesen en cada una de ellas (autonomía). La sintaxis jerárquica utilizada refleja la delegación de autoridad. Como un ejemplo, consideremos un espacio de nombres de la forma *local.sitio*, donde *sitio* es el nombre autorizado por la autoridad central, *local* es la parte del nombre controlada por el *sitio*, y el "." es el delimitador que separa a ambos. Sin embargo en un sistema jerárquico no sólo pueden existir un nivel de subdivisión, sino que pueden existir varios niveles de subdivisión. Un *sitio* puede consistir en varios grupos, y la autoridad del *sitio* puede elegir dividir su espacio de nombres según la cantidad de grupos que la componen (ejercicio de la autonomía delegada por la autoridad superior). La idea es subdividir el *sitio* de tal manera que este sea lo suficientemente manejable. Sintáxticamente introducimos otra partición al nombre. (por ejemplo: incorporamos una subdivisión *grupo* al *sitio, local.grupo.sitio*). Al *grupo.sitio*, se lo denomina domineo y a *local*, host o máquina, que se encuentra dentro del dominio. Un ejemplo concreto de direcciones Internet, es *ayelen.fisica.unlp.edu.ar* donde el sitio *ar* corresponde a la Argentina (toda máquina que pertenezca a la Argentina estará bajo la autoridad ar), *edu* corresponde a sitios académicos (universidades, laboratorios,...), *unlp* corresponde a un lugar geográfico como lo es la Universidad Nacional de La Plata, *física* corresponde al Dto. de Física de la UNLP y por último tenemos la máquina

denominada *ayelen* (nombre autóctono de la Argentina que identifica a la máquina dentro de física). En la dirección del ejemplo anterior ayelen.fisica.unlp.edu.ar, la máquina o host es ayelen y el domineo al que pertenece el host ayelen es fisica.unlp.edu.ar.

Ahora estamos en condiciones de introducirnos al formato de direcciones de los sistemas de mensajería dentro de un ambiente Internet. Como el nombre de la máquina y su interpretación de su composición ya estan resueltos, sólo nos faltaría ubicar el mailbox del usuario, pero con el nombre de este y haciendo uso del carácter @ daríamos por concluido la conformación general de la dirección del mensaje (por ejemplo: crusso@ayelen.fisica.unlp.edu.ar). En el ejemplo anterior "crusso" es llamado parte local de la dirección (local-part) según el RFC-822. La parte local es un string o secuencia de caracteres que será procesado oportunamente cuando este llegue a destino. Esto permite cierta flexibilidad para hacer redireccionamientos como veremos mas adelante. Un ejemplo podría ser *mailbox.subdominio1 @Este_dominio*, este es un caso de abreviaturas de referencias del subdominio local y de subordinar subdominios.

Explícitamente sobre Internet, uno puede si quiere dar una camino explícito para el recorrido de los mensajes de la siguiente manera <*ruta*>, donde *ruta* específica la secuencias de máquinas y/o servicios de transmisiones que han de atravesar (por ejemplo: < @dominio1 @dominio2 @dominio3:direccion>, léase vaya hasta *@dominio1*, pase por *@dominio2* y luego envíese el mensaje a *direccion*.

Direcciones en UUCP

El UUCP fue quizás uno de los primeros sistemas de transportes para el correo electrónico. Como hemos dicho el UUCP es un sistema de transferencia de archivos no interactivo. En otras palabras la conexión de una máquina a otra esta determinado por diferentes parámetros que son configurados por los systems managers de cada una de ellas, estos determinan hora y días que se han de conectar para transferirse información de una a otra (mail, archivos,...).
La confección de las direcciones de mensajería dentro de este protocolo de comunicaciones variaron con el correr de los años. En primer lugar las direcciones de los mensajes se confeccionaban según las máquinas que este debía usar para que el mensaje llegara a destino. Si un usuario desea mandar un mail a otro haciendo uso de máquinas intermediarias (sobre UUCP), este debe ponerlas Explícitamente en la dirección (por ejemplo: si el usuario A de la máquina X1 desea enviarle un mensaje al usuario B haciendo uso de la máquina X2 y luego de la máquina X3, entonces la dirección se compondría de la siguiente manera X2!X3!B, léase pase por la máquina X2 y por X3 y entréguese al mailbox B). En el ejemplo anterior la máquina X1 se conectaría con UUCP con la máquina X2 y le enviaría un mensaje con la dirección X3!B, luego la máquina X2 se conectaría con X3 y le enviaría un mensaje al usuario B que pertenece a dicho máquina. Genéricamente las direcciones sobre UUCP se verían como host1!host2!host3!usuario. Algunas configuraciones utilizan (en caso de no existir un enlace con la máquina remota) a UUNet como última alternativa de ruteo.

Notemos que a diferencia de Internet los usuarios en este protocolo no se abstraen de la configuración de la dirección de los mensajes. Ya que su

compocisión esta íntimamente relacionada con el éxito o no de que los mensajes lleguen a destino. Intuitivamente uno puede ver que la confección de las direcciones en UUCP no son jerárquicas ni únicas. No son jerárquicas debido a que las conexiones lógicas entre máquinas con UUCP podrían llegar hacer todas contra todas, y no es única fundamentalmente porque la dirección esta formada por el camino que ha de hacer el mensaje sobre dicha red de máquinas, con lo cual existen una gran variedad de posibilidades para que el mensaje llegue a destino.

Direcciones en ISODE (X400(84) - X400(88))

Ya examinamos anteriormente el sistema de direccionamiento de Internet sobre TCP/IP (user@domain - usuario@dominio). Este es el sistema de direccionamiento usado por ARPANET, la interconección de redes ARPA, USENET y otras. Los dominios pueden tener subdominios como ya lo hemos visto por lo que crusso@ayelen.fisica.unlp.edu.ar se refiere a Claudia Russo perteneciente al Dto. de Física de la UNLP mientras que crusso@info.unlp.edu.ar se refiere a Claudia Russo perteneciente al Dto. de Informática de la UNLP. Este sistema de nomenclatura basado en dominio funciona adecuadamente con un conjunto de usuarios de aproximadamente un millón de personas, en las universidades, en empresas corporativas, en oficinas gubernamentales,... . Sin embargo la International Consultative Committee for Telegraphy and Telephony (CCITT) no piensa que funcionará tan adecuadamente, cuando el correo electrónico abarque a la totalidad de la población (hoy según un estudio realizado por Intel, sólo el 2% de la población mundial tiene acceso a computadoras), por lo cual se dedicó a diseñar un esquema de direccionamiento radicalmente distinto (comparado con los de TCP/IP y UUCP), conjuntamente con la ISO.

Los teléfonos, los equipos de télex y otras entidades direccionables, que se encuentran en las redes de computadoras, tienen siempre identificadores numéricos largos, para asegurar que no haya ninguna ambigüedad en la identificación. Dado que la gente tiende a olvidar los números, se proporcionan guías telefónicas, donde se correlacionan los nombres con los números telefónicos. Todo converge a que se reemplace las guías telefónicas por servicios de consulta por computadora llamado servicio de directorio en línea, que proporcionará información telefónica, direcciones de redes,... . Para evitar caos la ISO y el CCITT están desarrollando normas para este servicio de directorio. A simple vista, parecería suficiente proporcionar una simple lista alfabética de nombres, como la que provee el servicio telefónico argentino a sus usuarios. El sistema de identificación alfabética que tienen los argentinos con sus guías varían de país en país. Por ejemplo: los directorios telefónicos de todo el mundo se clasifican según el apellido, aunque precisamente lo que constituye el apellido varia de país en país. En la Argentina toda la gente con el mismo apellido queda clasificada mediante el nombre como clave secundaria; en Holanda la dirección de la calle forma la clave secundaria.

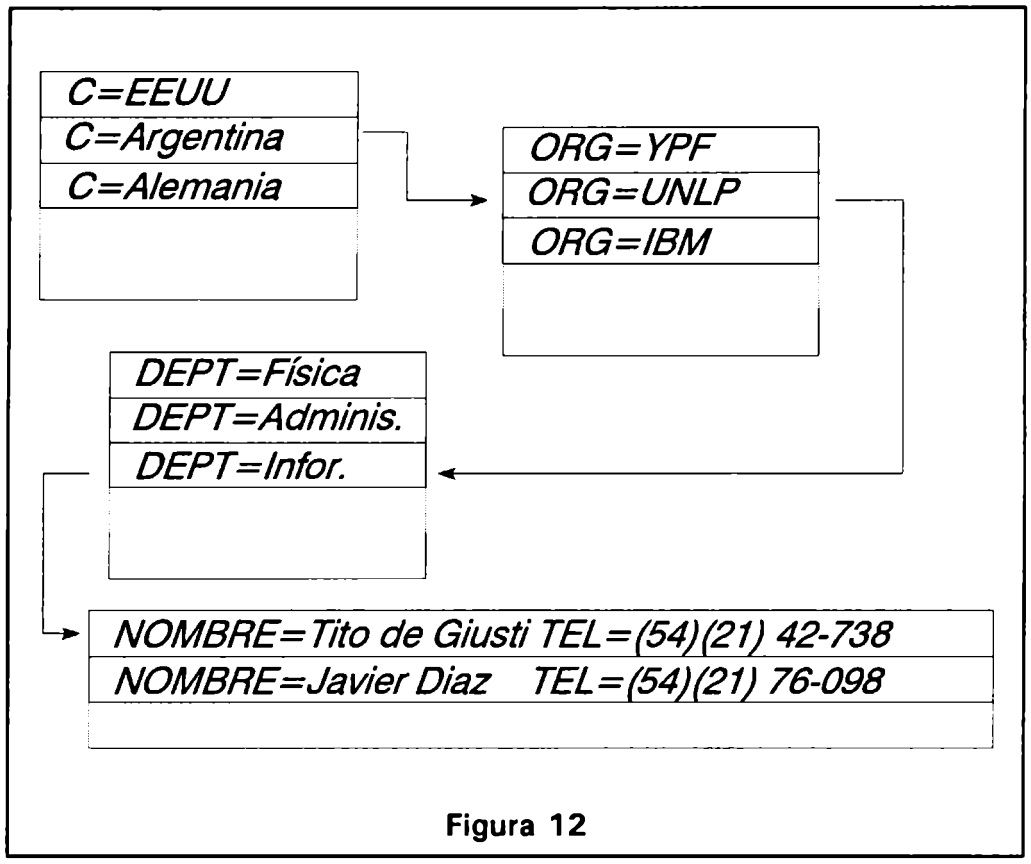

Figura 12

La idea básica del servicio de directorio de OSI (llamado por CCITT como X500) es permitir que los usuarios busquen los nombres basándose en atributos (por ejemplo: alguien podría tratar de averiguar el número telefónico de Tito de Giusti, que trabaja en la UNLP de la ciudad de La Plata). Para tomar en cuenta las convenciones de los distintos países , así como los servicios privados de los directorios dentro de las empresas, los servicios de directorios estan compuesto implícitamente en forma jerárquica. Aquí, el nivel superior posee registros de la forma *C=país*. Un nivel más abajo, podemos observar identificadores de organizaciones, llamados *ORG=nombre_de_la_organización*. Debajo de ese nivel tenemos departamentos y, en la parte inferior, entradas para la gente (o servicios, máquinas, impresoras,...). Los atributos del nivel inferior se pueden heredar, de tal manera que todos los registros bajo *ORG = YPF* actuarán, como si tuvieran también el atributo *C=Argentina*.

El formato de dirección que adopto la ISO (en consecuencia aplicadas a ISODE) estan especificadas en las recomendaciones de CCITT X400(84) y X400(88), esta última especificada en forma conjunta ISO con CCITT. (84 y 88 fueron los años 1984 y 1988 en el que fueron presentadas). Estas direcciones han sido diseñadas como un conjunto de atributos, donde cada atributo tiene un tipo y un valor (por ejemplo: el atributo Nombre podría tener como tipo ASCII, y como valor de ese atributo podría ser Adrián). Los atributos estan subdivididos en dos tipos, el

primero es llamado Atributos Estándares (Standars Attributes - SA's) y el segundo son los Atributos Definidos de Dominio (Domain Definied Attributes DDA's).

Los SA's, son aquellos atributos que todas las instalaciones de X400 reconocen y soportan (por ejemplo: nombre de país, unidad organizacional,...). Los atributos mas frecuentemente usados dentro de los atributos estándares son:

SurName (S)
GivenName (G)
Initials (I*)
GenerationQualifier (GQ)
OrganizationalUnits (OU1 OU2 OU3 OU4)
OrganizationName (O)
PrivateDomainName (PRMD o P)
AdministrationDomainName (ADMD o A)
CountryName (C)

La combinación de los atributos S, G, I* y GQ es comúnmente referenciada como PersonalName (PN).

No existe ninguna jerarquía definida por el estándar, sino que esta dada implícitamente en forma natural:

C>A>P>O>OU1>OU2>OU3>OU4>PN

De los SA's que se listaron, se incorporó uno llamado CommonName (CN). El CN puede ser usado sólo o conjuntamente con PN. Unos de los problemas de X400(84) fue que el PN fue creado para representar personas, pero no roles o objetos abstractos, como una lista de distribución (por ejemplo: en X400(84) la dirección del postmaster se incorporaba al atributo S (*S=postmaster*) cuando el postmaster en realidad es un rol; en X400(88) se salvo esa diferencia con el atributo CN, *CN=postmaster*).

Los DDA's pueden ser usados como agregados de los SA's. Una instancia de un DDA consiste de un tipo y su valor correspondiente. Los DDA's a veces tienen un significado dentro de determinado contexto (por ejemplo: una compañía podría querer definir un DDA para describir los diferentes teléfonos internos que posee, DDA type = phone value = 21-1889).

Los atributos C y ADMD, deben estar presentes en una dirección y no pueden estar vacíos. También almenos uno de los atributos PRMD, O, OU, PN y CN deben estar presentes. Las direcciones están definidas en X400 usando Abstract Syntax Notation One (ASN.1 - X409), aquellas que sólo poseen SA's son llamadas Standard Attribute Addresses (SAA's). Hasta ahora hemos hablado de los atributos que componen las direcciones y no de su formato. Aclaremos que X400 estándariza la representación de la codificación binaria de la dirección, este no estándariza como la dirección será escrita sobre papeles o como la verá el usuario sobre la pantalla de la computadora. Existen algunas recomendaciones de CCITT para los formatos de representación visual de las direcciones de X400. A continuación se presentarán tres formatos de direcciones en X400, conformados con los atributos que hemos expuesto anteriormente.

Se propuso un anexo a CCITT y a ISO llamada *Representation of O/R addresses for human usage*. Según esta recomendación las direcciones en X400 se verían así:

G = Claudia Cecilia; S = Russo; O = Exactas; OU1 = Informática; OU2 = LIDI; P = UNLP; A = Secyt; C = Arg.

Observe que en este formato, el orden del atributo O y OU es exactamente lo opuesto a lo que uno espera intuitivamente (los atributos estan puesto en un orden jerárquico de izquierda a derecha, menos en los atributo O y OU respectivamente, donde en estos esta puesto de derecha a izquierda). Algunas de las EAN's (Europe Academic Network) utilizan esa propuesta con algunas modificaciones. Se solucionó el problema jerárquico, no numera los atributos OU (ya que si el formato es jerárquico es redundante numerarlos) y permite escribir en la dirección ADMD y PRMD en ves de A y P. Entonces siguiendo con el ejemplo anterior, nos quedaría:

G = Claudia Cecilia; S = Russo; OU = LIDI; OU = Informática; O = Exactas; P = UNLP; A = Secyt; C = Arg.

Otro formato es el expuesto en el RFC-1327, que define el carácter "/" como separador de atributos (es el que usaremos más adelante, cuando hablemos de direcciones X400 en PP). Siguiendo siempre sobre el mismo ejemplo nos quedaría:

/G = Claudia Cecilia/S = Russo/OU = LIDI/OU = Informática/O = Exactas /P = UNLP /A = Secyt /C = Arg/

Clasificación de direcciones según "Lennart Lövstrand"

La clasificación de direcciones que hemos expuesto en este documento se baso en explicar los orígenes y fundamentos sobre los diferentes transportes que utilizan los MTA's para el envío o reenvío de mensajes. Lennart Lovstrand realiza otra interesante clasificación que vale la pena comentar, en un paper de su autoría titulado *Electronic Mail Addressing in Theory and Practice*, donde las clasifica según su formato, con el cual genera cuatro grupos que son:

Direcciones Relativas

Son aquellas direcciones que señalan la ruta por el cual el mensaje debe pasar hasta llegar al destino (por ejemplo: las direcciones UUCP y las direcciones de Internet de la forma < @domineo,@domineo:dirección >).

Direcciones Absolutas

Son aquellas direcciones que son universalmente únicas aplicables por cualquier MTA independientemente donde este ubicado (por ejemplo: las direcciones de Internet *mailbox@domain*)

Direcciones con atributos

Es el caso de las direcciones de X400 ya vistas en este informe.

Direcciones Híbridas

Son aquellas direcciones que son una mezcla de las tres anteriores. Sobre este tema hablaremos a continuación.

Problemas de direccionamiento

Aprovecharemos la clasificación hecha por Lennart para introducir dentro de las direcciones híbridas una problemática que hoy en día deja mucho de que hablar. Concretamente hoy existen en el mundo dos grandes especificaciones estándares de formatos de mensajes que abarcan la mayoría de la correspondencia electrónica, ellos son el RFC-822 y el X400(84/88) (que hemos comentado en párrafos anteriores). Sin embargo en Europa (lugar donde el X400 tiene mayor aceptación que el RFC-822) quieren intercambiar mensajes con usuarios que utilizan formato RFC-822 (por ejemplo: con los EEUU, donde el RFC-822 prevalece sobre el X400). Esto da inicio a pensar en adaptar los MTA's para que actúen como sistemas de transformación de mensajes de un mundo a otro (recibir un mensaje en X400 y transformarlo a RFC-822 de la misma manera que recibir un mensaje en RFC-822 y transformarlo a X400). Estos nuevos MTA's son llamados *gateways*. Como ya hemos visto las direcciones X400 y las especificadas en el RFC-822 no coinciden en su filosofía, con lo cual se generan algunos inconvenientes en la transformación de un formato a otro. Lo mismo ocurre con sistemas de mensajería basados en transportes como el UUCP. Si bien utilizan el estándar RFC-822, el formato de direcciones es filosóficamente distinto al de Internet.

Citaremos ejemplos concretos con sus explicaciones pero no detallaremos como son las reglas de transformación ya que no es el objetivo de este trabajo.
Es muy común ver direcciones híbridas entre UUCP e Internet de la forma *maquina!maquina!usuario*@*domineo*. Si analizamos el ejemplo *maquina!maquina!usuario* será visto como parte local que será evaluada cuando se llegue a @*domineo*, quien la enviara por UUCP a la dirección *maquina!maquina!usuario* (observe que el carácter "@" tiene precedencia sobre el carácter "!").

Es frecuente ver en gateways entre RFC-822 y X400, direcciones de la forma *"direccion_en_formato_X400"*@*domineo*, donde se intentará llegar primero a @*domineo* quien luego de pasar al mundo X400 usará la dirección especificada en *direccion_en_formato_X400* (el uso del carácter " sirve para salvar cualquier ambigüedad que pudiese ocurrir entre el formato de direcciones X400 y RFC-822).

Message Transfer Agents (MTA's)

Introducción

En párrafos anteriores introducimos superficialmente algunos servicios y conceptos que ofrecen los MTA's a los UA's. Existen otros servicios y conceptos que serán abarcados a continuación a medida que vayamos viendo más detenidamente cada uno de los MTA's en estudio (Sendmail, MMDF y PP).

Abordaremos en principio al Sendmail, este MTA es en la actualidad el más difundido en todo el globo terráqueo en lo que a MTA se refiere, básicamente por ser un software de dominio público y por ser uno de los precursores. Luego se describirán otras alternativas con diferentes características.

Sendmail

Introducción

El Sendmail tuvo sus orígenes en la Universidad de Berkeley (EEUU) a mediados de la década del 70 (ya mencionamos en párrafos anteriores que en esa época reinaba la anarquía en los sistema de mensajería). Se llamó en sus comienzos *delivermail*. Sirvió como enlace de correo entre la Universidad de Berkeley y el mundo. Desafortunadamente no era lo suficientemente flexible para adaptarse a los cambios que se producían durante esa época, sin embargo este software aparece en las versiones de BSD UNIX 4.0 y 4.1.

A los inicios de 1980 ARPANET cambió su protocolo NCP (Network Control Protocol) a TCP (lo que permitió que no sean 256 máquinas las conectadas sino millones de ellas), sumado a esto la aparición del SMTP, la jerarquías de nombres sobre TCP/IP, llevó al creador del delivermail (Eric Allman) a transformarlo en lo que hoy llamamos Sendmail.

A medida que incursionemos sobre este MTA no nos olvidemos en la época en que fue diseñado. Recordemos que su etapa de desarrollo y evolución fue una etapa de bombardeos y cambios bruscos que se producían sin previa anticipación, esto condujo a Eric Allman a buscar una forma de adaptación ágil y segura. Hoy este MTA es mundialmente reconocido, empresas de renombre internacional (Hewlett Packard, SUN, Silicon Graphics, SCO, Digital,...) lo han adoptado como MTA de base y así también aquellos audaces que han desarrollado sistemas operativos Unix de dominio público (Linux,...). El Sendmail es más que un simple programa, es una colección de programas, archivos, directorios y servicios.

Cómo se definen los diferentes transportes que se han de usar ?. Cómo el Sendmail selecciona los transportes para hacer la entrega del mensaje?. Cómo se define los alias del sistema y cómo se manejan las listas de usuarios?. Que son las reglas que utiliza el Sendmail y para que sirven?. Cómo trabajan esas reglas?. La clave de estos y otros interrogantes se encuentran en un archivo de configuración llamado *sendmail.cf*, más aún, la filosofía de desarrollo de cada uno de los MTA's se encuentran reflejadas en cada uno de sus archivos de configuración. Este documento nos enseñará cada una de sus partes, operatividad, funcionalidad y aplicabilidad de estos archivos. En particular en el sendmail.cf esta definido en base a un conjunto de reglas sintácticas que serán introducidas en breve a medida que vayamos abordando los diferentes temas que involucran a los MTA's. Inicialmente

definiremos los aspectos operacionales y funcionales de los MTA's, para luego abarcar la tabla comparativa y las conclusiones en el próximo capítulo.

Definición de transportes

El programa más importante del conjunto de ellos es el *sendmail* propiamente dicho. Este es ejecutado (según una serie de flags) en modo background (como un deamon) a la escucha de mensajes sobre aquellos transportes (LOCAL, UUCP, TCP/IP, FAX,...) que le han sido configurados (sendmail -bd, el flag -bd indica que correrá como deamon). Los transportes están definidos en el archivo de configuración (sendmail.cf). Trabajaremos con un ejemplo para hacer la explicación más didáctica.

***Mtcp**, P=[IPC], A=IPC $h*
***Mlocal**, P=/bin/mail, F=lsDFMmnP, S=10, R=20, A=mail -d $u*
***Muucp**, P=/bin/uux, F=lsDFMmnP, S=13, R=23, A=uux - -r $h!rmail ($u)*

El carácter ***M*** al comienzo de la línea nos indica que vamos a definir un transporte sobre el cual recibiremos y enviaremos mails. El sendmail no efectúa la entrega de correspondencia, de aquí se deduce que el servicio local deberá ser ejecutado a través de un transporte (en nuestro ejemplo ese transporte es llamado *local*). Los otros transportes definidos *tcp* y *uucp* son los protocolos de comunicaciones TCP/IP y UUCP respectivamente.

El carácter ***P*** nos indica el programa que realizará en definitiva la entrega efectiva del mensaje.

*Mtcp, **P=[IPC]**, A=IPC $h*
*Mlocal, **P=/bin/mail**, F=lsDFMmnP, S=10, R=20, A=mail -d $u*
*Muucp, **P=/bin/uux**, F=lsDFMmnP, S=13, R=23, A=uux - -r $h!rmail ($u)*

Observemos la definición del transporte tcp, la variable *P* no indica en ese caso un programa. Esto es así porque el transporte sobre este protocolo esta configurado internamente (dentro del sendmail). La pregunta que nos podemos hacer es porque sobre este protocolo en particular esta internamente configurado y los otros no?. La respuesta es sencilla, el protocolo usado sobre TCP/IP para servicios de mensajería es el SMTP, como veremos más adelante este protocolo es un protocolo interactivo lo opuesto del */bin/mail* y del */bin/uux*.

El carácter ***A*** especifica la línea de argumentos que ha de ejecutarse a la hora de la entrega de los mensajes (note que el primer argumento es el nombre del programa ha ejecutarse).

*Mtcp, P=[IPC], **A=IPC $h***
*Mlocal, P=/bin/mail, F=lsDFMmnP, S=10, R=20, **A=mail -d $u***
*Muucp, P=/bin/uux, F=lsDFMmnP, S=13, R=23, **A=uux - -r $h!rmail ($u)***

En el caso del transporte tcp la línea de comando sufre la misma característica que en el caso anterior.

El carácter ***F*** contiene una serie flags que le dicen al sendmail más acerca de la forma de entrega del mensaje.

Mtcp, P=[IPC], A=IPC $h
*Mlocal, P=/bin/mail, **F=lsDFMmnP**, S=10, R=20, A=mail -d $u*
*Muucp, P=/bin/uux, **F=lsDFMmnP**, S=13, R=23, A=uux - -r $h!rmail ($u)*

Cada una de estas letras actúa como una variable booleana (si esta presente es verdadero y sino es falso). Por ejemplo el flag *F* nos indica que el campo From: del RFC-822 debe estar presente en el encabezamiento, al igual que el flag *M* también nos indica que el campo Message-Id: también debe estar presente. Hablaremos de los restantes cuando tratemos temas relacionados a estos flags.

Las letras ***S*** y ***R*** nos indican las reglas de sobre escritura de la dirección del que envía (sender) y del que recibe (recipient). Estas reglas serán explicadas a continuación.

Existen otras variables que permiten configurar cual ha de ser el máximo mensaje que ha de poder pasarse a través de él (por ejemplo: M=200000) y cual ha de ser la máxima línea que se permite transferir (por ejemplo: L=80).

Macros

El Sendmail cuenta con la posibilidad de definir macros. La macro puede verse como una variable, donde tenemos el nombre de la variable (de un sólo carácter) y su contenido. Veamos el siguiente ejemplo *Dwayelen.fisica.unlp.edu.ar* El carácter *D* nos está indicando que vamos a definir una macro. El carácter *w* el nombre de la macro y *ayelen.fisica.unlp.edu.ar* es el contenido de *w*. Para acceder al contenido se lo hace haciendo uso del carácter *$* (por ejemplo: $w).
El uso de macros permite mayor versatilidad y modificabilidad al archivo de configuración. Existen macros del sistema ya predefinidas (ejecutando el comando *sendmail -d35.9 -bt < /dev/null* Ud las podrá ver). El nombre de la macro debe tener un solo carácter, esa restricción tiene el inconveniente de que no son nemotécnicas lo que dificulta en cierta manera una rápida comprensión del contenido del archivo de configuración, pero posee la ventaja de ser más eficiente cuando internamente el sendmail hace uso de ella.

Clases

Existen macros cuyos contenidos no contienen un único valor sino una lista de ellos, en esos casos se utilizan las clases. La manera de declarar una clase es haciendo uso del carácter *C*, a continuación va el nombre (al igual que la macro el nombre es de un sólo carácter) y luego sus valores separados por caracteres en blanco (por ejemplo: *Cw ayelen ayelen.fisica*)

Opciones

Las opciones son precedidas con el carácter *O*, luego el nombre de la opción y a continuación el contenido. Estas variables configuran de alguna manera al Sendmail determinando la ubicación física de la cola de trabajo (por ejemplo: OQ/usr/spool/mqueue), los tiempos en que un mensaje estará a lo sumo encolado

(por ejemplo:OTd5 como máximo 5 días), los permisos de los archivos (por ejemplo:OF0600), el usuario y grupo por defecto (por ejemplo:Ou1, Og1),...

Prioridades

Las prioridades determinan si un mensaje ha de ser procesado antes que otro o no. Es muy común que las listas tengan baja prioridad, para una mayor eficiencia en la administración. Las prioridades son identificadas con el carácter *P* y su sintaxis es *Pnombre = valor*.

Pspecial-delivery = 100	*Se utiliza para cuando se esta haciendo una transferencia efectiva*
Pfirst-class = 0	*Cualquier mensaje estándar (por defecto)*
Plist = -30	*Para mensajes originados como parte de una lista*
Pbunk = -60	*Broadcast (menos importante que una lista)*

Reglas

Un usuario hace uso del correo electrónico en principio desconociendo cual ha de ser el transporte que se ha de utilizar para que el mensaje llegue a destino. Es responsabilidad del Sendmail determinar cual ha de ser el trasporte ha utilizar. Este principio esta reflejado en la siguiente gráfica.

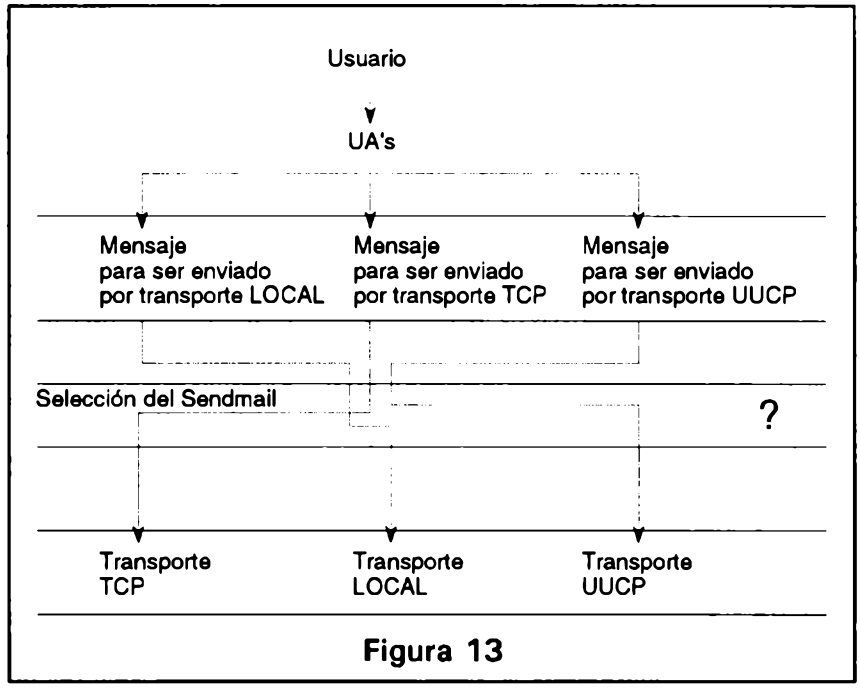

Figura 13

El Sendmail usa los formatos de direcciones para determinar el transporte a utilizar. Esas direcciones son procesadas por el Sendmail a través de reglas que son definidas por el administrador del sistema de correo. Estas reglas sirven para modificar las direcciones, detectar errores de direccionamiento y por último seleccionar medios de transportes. A continuación explicaremos de que se tratan esas reglas, como se aplican, cual es sus sintaxis,....

Area de trabajo (Workspace)

Antes de hablar de las reglas, hablemos de como trabaja internamente el sendmail. El sendmail posee un buffer donde procesa las direcciones. Ese buffer llamado workspace contiene la dirección a procesar, cada una de las componentes de ese buffer es llamada token, por ejemplo:

buffer o workspace "crusso" "@" "lidi" "." "unlp" "." "edu" "." "ar"

crusso	*token 1*	***unlp***	*token 5*	*ar*	*token 9*
@	*token 2*	*.*	*token 6*		
lidi	*token 3*	***edu***	*token 7*		
.	*token 4*	*.*	*token 8*		

Observe que la dirección crusso@lidi.unlp.edu.ar está clasificada por separadores "." y "@" (los separadores que intervienen están definidos en una macro de la siguiente forma Do.:%@!^=/[ ]).

Definición de una regla

Una regla puede estar definida por un conjunto de reglas o en particular por ninguna. Se utiliza el carácter *S* para dar inicio a la declaración de la regla que termina cuando se encuentra con otro carácter *S*. A continuación de la *S* va el N° de la regla (desde 0 a 6 son internamente definidas por el sendmail para propósitos específicos). Para las reglas internas dentro de una regla se utiliza el carácter *R*. Veamos el siguiente ejemplo:

S0			*Inicio de la regla 0*
Rlhs	*rhs*	*comentario*	*regla perteneciente a S0*
.....			
Rlhs	*rhs*	*comentario*	*regla perteneciente a S0*
Sn			*Fin de la regla S0 y comienzo de la regla n*

El modo de evaluar las reglas R es verla como un *if-then-else*, es decir si *lhs* (left hand side) es verdadero entonces hacer *rhs* (right hand side) sino continuar con la regla inmediata inferior. Cómo se evalúa el valor de verdad del lhs?. En el workspace tenemos descompuesta la dirección a procesar según los separadores definidos. Si esta dirección machea con el lhs entonces el lhs es verdadero, y es ahí donde la dirección que se encuentra en el workspace es sobre escrita por el rhs. Veamos el siguiente ejemplo (observe que obviamos la separación que hace el workspace con los tokens).

Ejemplo 1

Estado inicial del workspace	*local@aqui*	
Regla a analizar	*Rayelen@aqui*	*no machea*
Estado final del workspace	*local@aqui*	

Ejemplo 2

Estado inicial del workspace	*local@aqui*		
Regla a analizar	*Rlocal@aqui*	*root@ayelen*	*machea*
Estado final del workspace	*root@ayelen*		

Analicemos el flujo sobre un conjunto de reglas en base a un ejemplo:

S0 *contenido del workspace = x*
Rx *y*
Ry *z*

Si se ejecuta la regla S0 a través de las reglas que la componen (Rx, Ry), el contenido final del workspace sería z. Al comenzar, la regla *Rx* machea con el workspace con lo que su contenido se sobrescribirá con el valor *y* continuará evaluando la regla siguiente, pero nuevamente el workspace machea con la regla *Ry* entonces nuevamente se sobrescribe con el valor *z* que es el valor del retorno.

Operadores de las reglas

Una vez introducido el control de flujo de las reglas nos concentraremos en los operadores que intervienen en su construcción. Los operadores se clasifican como operadores del lhs y como operadores del rhs. Las tablas que están a continuación nos mostrarán todos los operadores definidos en el Sendmail con una explicación breve de su funcionamiento.

Operadores lhs	*Significado*
*$**	*Machea cero o más tokens*
$+	*Machea uno o más tokens*
$-	*Machea exactamente un token*
$=x	*Machee cualquier token o concatenación de tokens en la clase x*
$~x	*Machee un único token no de la clase x*

Operadores rhs	*Significado*
$h	*Inserta un token en la reescritura de la dirección*
$>n	*Llama a la regla n*
$<programa	*Llama a un programa para reescribir la dirección*
$[máquina$]	*Canoniza la máquina*
$#transporte	*Determina el transporte a utilizarse*
$@	*Sirve para retornar la máquina o como retorno inmediato de una regla*
$.	*Retorna el usuario*

Los operadores lhs determinan su valor de verdad en un proceso llamado *pattern matching*. Trataremos en donde sea posible dar la funcionalidad de los operadores con ejemplos.

Operador $* (lhs)

Ejemplo 1

Regla		
*R$**	*rhs*	*regla a aplicar*
workspace	***pattern***	
(vacio)	*$**	*machea con cero o más tokens*

Ejemplo 2

Regla		
*R$*@$**	*rhs*	*regla a aplicar*
workspace	***pattern***	
"root"	*$**	*machea con cero o más de un token*
"@"	@	*machea exactamente el token*
"unlp"	*$**	*machea con cero o más de un token*
workspace	***pattern***	
"root"	*$**	*machea con cero o más de un token*
"@"	@	*machea exactamente el token*
	*$**	*machea con cero o más de un token*
workspace	***pattern***	
	*$**	*machea con cero o más de un token*
"@"	@	*machea exactamente el token*
	*$**	*machea con cero o más de un token*

Ejemplo 3

Regla		
*R$*@$**	*rhs*	*regla a aplicar*
workspace	***pattern***	
root	*$**	*machea con cero o más de un token*
	@	*no machea*
	*$**	

por lo tanto se sobre escribe el workspace con el rhs tanto en el primer ejemplo como en el segundo y en el tercero al no machear se tratará de seguir con el flujo de ejecución como se describió anteriormente.

Operador $+ (lhs)

Ejemplo 1

Regla		
R$+	*rhs*	*regla a aplicar*
workspace	***pattern***	
"root"	*$+*	*machea con almenos un token*
"@"		*y opcionalmente con varios*
"unlp"		

Ejemplo 2

Regla		
R$+@$+	*rhs*	*regla a aplicar*
workspace	***pattern***	
"root"	*$+*	*machea con almenos un token*
"@"	@	*machea exactamente el token*

"unlp" *$+* *machea con almenos un token*

por lo tanto se sobre escribe el workspace con el rhs tanto en el primer ejemplo como en el segundo.

Operador $- (lhs)

Ejemplo 1

Regla

R$-@$+ *rhs* *regla a aplicar*

workspace ***pattern***

"root" *$-* *machea con exactamente uno*

"@" *@* *machea exactamente el token*

"unlp" *$+* *machea con almenos un token*

Ejemplo 2

Regla

R$-@

workspace ***pattern***

$- *no machea con exactamente uno*

"@" *@* *machea exactamente el token*

"unlp"

por lo tanto se sobre escribe el workspace con el rhs en el primer ejemplo y en el segundo tratará de continuar con el flujo de ejecución.

Operador $ = x (lhs)

Ejemplo

Clase

Cw localhost localhost.dominio

Regla

R$-@$=w *rhs* *regla a aplicar*

workspace ***pattern***

"usuario" *$-* *machea exactamente un token*

"@" *@* *machea exactamente el token*

"localhost" *$=w* *machea con un token del clase w (localhost.dominio)*

"."

"dominio"

por lo tanto se sobre escribe el workspace con el rhs.

Operador $ ~ x (lhs)

Este operador es la inversa del operador $ = .

Ejemplo

Clase

Cl intruso1 intruso2

Regla

R$-@$~l *rhs* *regla a aplicar*

workspace ***pattern***

"usuario" *$-* *machea exactamente un token*

"@" *@* *machea exactamente el token*
"intruso1" *$=w* *machea exactamente con un token del clase I (intruso1)*

por lo tanto se sobre escribe el workspace con el rhs.

Operador $n (rhs)

Ejemplo 1

Clase
Cw localhost localhost.dominio
Regla
R$-@$=w *$1.localhost* *regla a aplicar*
1 2 3
workspace ***pattern***
"usuario" *$-* *machea exactamente un token*
"@" *@* *machea exactamente el token*
"localhost" *$=w* *machea con un token del clase w (localhost.dominio)*
"."
"dominio"
resultado
"usuario" "@" "localhost"

Observe que esta regla, lleva cualquier dirección local a la forma usuario@localhost. Al machear el workspace con el lhs de la regla entonces el workspace se sobre escribe con $1, donde $1 es el contenido que ha macheado con el operador $- ($2 es @ y $3 es el contenido de $=w).

Operador $> (rhs)

Este operador llama a otra regla pasándole como parámetro el contenido del workspace y retornando un valor que en definitiva será el que contendrá el workspace como valor de retorno.

Operador $<programa (rhs)

Este operador nos permite ejecutar ciertos programas con el fin de encontrar determinados valores, para sobre escribir las direcciones. Estos permiten acceder a name servers. Programa es el nombre de una macro que contendrá el nombre del programa sin argumentos. La dirección que machea en el lhs de la regla reescrita por el rhs de la regla es pasada como argumento simple al programa definida en la macro.

Ejemplo

Macro
DP/usr/bin/uupath
Regla
R$+<@$-> *$:$<P1@$2* *regla a aplicar*
1 2
workspace ***pattern***
"usuario" *$+* *machea almenos un token*
"<" *<* *machea exactamente el token*
"@" *@* *machea exactamente el token*

"localhost" *$-* *machea exactamente un token*
resultado
el valor de retorno de "/usr/bin/uupath usuario@localhost"

Operador $[máquina$]

Este permite, si el name server esta en uso, reemplazar una dirección de Internet (IP) por su nombre (máquina.dominio) o su nombre (máquina) por el nombre completo (máquina.dominio).

Ejemplo

Regla
R$*<@$+>$** *$:$1<@$[2]>$3 *regla a aplicar*
workspace ***pattern***
"usuario" *$** *machea almenos un token*
"<" *<* *machea exactamente el token*
"@" *@* *machea exactamente el token*
"localhost" *$+* *machea exactamente un token*
">" *>* *machea exactamente el token*
*$** *machea cero tokens*
resultado
el valor de retorno del name server de localhost (por ejemplo: localhost.dominio),
"usuario" "<" "@" "localhost" "." "dominio" ">"

Operador $#transporte, $@, $: (rhs)

Retorna el trasporte ha ser utilizado para la entrega o reenvío del mensaje. Este operador trabaja con una terna de valores que son el transporte ha usarse, la máquina al cual deseo llegar (operador $@) y el usuario al que tengo que enviárselo (el operador $:).

Ejemplo

Regla
R$+@$+ *$#local$@2:$1* *regla a aplicar*
workspace ***pattern***
"usuario" *$** *machea almenos un token*
"@" *@* *machea exactamente el token*
"localhost" *$+* *machea exactamente un token*
resultado
El transporte es "local", la máquina es "localhost" y el usuario es "usuario" .

Otra función del operador $@ (rhs)

Si el rhs comienza con $@ entonces se sobrescribe el workspace en función del rhs si se cumple el lhs y automáticamente se retorna el contenido del workspace.

Funcionalidad de las reglas

Las reglas que van desde la 0 a la 6 tienen propósitos bien definidos, que se muestran en la tabla que esta a continuación.

Regla	**Función**
S0	La regla S0 determinará la terna transporte, dirección de la máquina destino y usuario (#transporte $@máquina $:usuario)

S1	Procesa la dirección del emisor, bajo los criterios que el postmaster decida en la especificación de cada una de las reglas que componen a la regla S1.
S2	Idem a S1, pero para la dirección del receptor.
S3	Esta regla sirve como preprocesamiento de las direcciones, elimina texto que bajo cierto contexto no tiene ningún significado.
S4	Preprocesa todas las direcciones.
S5	Preprocesa el encabezamiento del emisor.
S6	Preprocesa el encabezamiento del receptor.

Explicación sintáctica del uso de las reglas

Las reglas se utilizan para el procesamiento de las direcciones. El flujo de la ejecución de las reglas se ve en la figura que está a continuación.

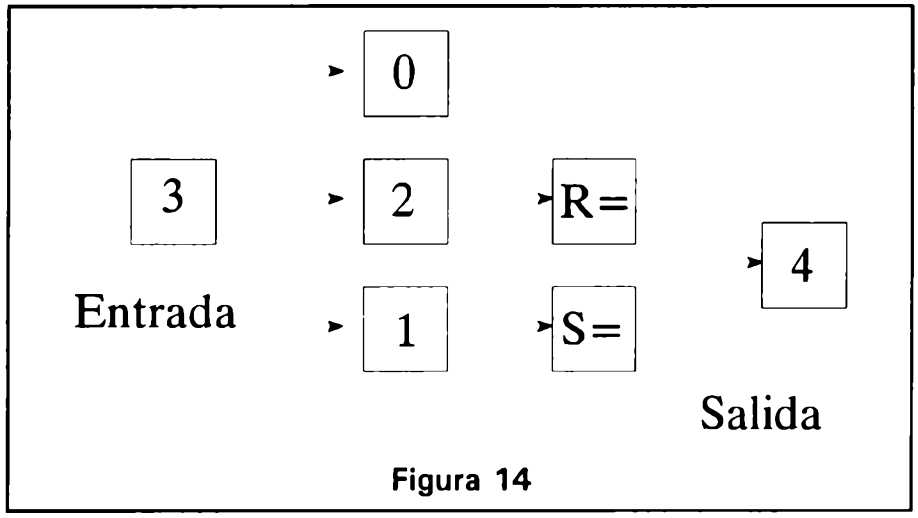

Figura 14

Esta gráfica trata de resumir de alguna manera el control de flujo de las reglas. Para dar una explicación más acabada del uso, daremos un ejemplo de la aplicabilidad de la regla 3. La regla 3 nos sirve como preprocesamiento de la dirección. Como bien sabemos las direcciones en RFC-822 vienen acompañadas de texto adicional que informa al lector del o los nombres de quienes han enviado el mensaje (por ejemplo: From: Luis Alberto Rizzi <rizzi@cetad.unlp.edu.ar>). La regla 3 tiene por finalidad filtrar esa información que no tiene ningún sentido para el procesamiento posterior. También sirve para detectar errores de direccionamiento (por ejemplo: < >) Haciendo uso de las reglas podemos definir la regla 3 de la siguiente manera.

S3

R$ < > $** *$ > 0* *redirecciono a la regla 0 para que lo resuelva*

R$*<$>$* $@<$2> filtro.*

El significado *R=* y *S=* significa que se aplicará la regla 1 y/o 2 y luego continuará con la regla que se ha configurado en cada uno de los transportes con la letra *S* y/o *R*, para finalizar con la regla 4.

En este informe se presentará el archivo de configuración con el cual hemos trabajado en la HP-UX 9000/710 y Ud con los elementos que le hemos provisto más el flujo de ejecución de las reglas podrá evaluarlas. Este informe no hará una detallada explicación de las reglas implementadas. La complejidad de su explicación mas todas las variantes que intervienen son demasiadas variables para complicar este informe cuyo objetivo es la funcionalidad de este producto y no su eficiente operatividad. Con la breve explicación de la regla 3 quisimos dar una idea superficial de como se aplican los operadores para determinar los transportes, errores, filtros,... .

Encabezamiento (headers)

El Sendmail tiene la capacidad de configurar los encabezamientos en función de variables que se encuentran definidas en los transportes. La manera de modificar la configuración por omisión del Sendmail es haciendo uso de la letra *H*. Recordemos que cuando definíamos los transportes, utilizábamos la letra *F* para determinar cuales eran las características que le incorporábamos al mensaje. La sintaxis utilizada es la siguiente *H?flag?nombre: valor* lo que significa que si el *flag* está, entonces *nombre* tendrá el *valor*. Veamos el siguiente ejemplo.

Muucp, P=/bin/uux, F=lsDFMmnP, S=13, R=23, A=uux - -r $h!rmail ($u)
H?M?Message_ID: <$t.$i@$j>

$i y $t ($i identificador único de la cola de trabajo y $t día en formato entero de creación) son macros que poseen valor cuando es procesado el mensaje y $j es la macro que esta definida con el nombre la máquina. Un ejemplo completo sobre un encabezamiento del estándar RFC-822 haciendo uso del sendmail y de los flags de los trasportes podría ser:

#Headers
HFrom: $q
HReceived: by $j id $i; $b
H?x?Full-Name: $?x$x$.
H?D?Date: $a
H?M?Message-Id: <$t.$i@$j>

$q es una macro que contiene el valor del autor del mensaje, $b contiene el día de recepción del mensaje en formato estándar RFC-822, la macro $x contiene el nombre completo del quien lo envía.

Manejo de Alias

El manejo de alias se realiza a través de un archivo. Este archivo contendrá el alias y su valor correspondiente. Esos valores pueden ser direcciones de correo electrónico o otros alias. Existe una opción en el sendmail.cf informándole

al Sendmail donde esta el archivo de alias (por ejemplo: OA/usr/lib/aliases). Veamos el siguiente ejemplo:

root <Adrian Russo>: arusso@ayelen.fisica.unlp.edu.ar, crusso
crusso: crusso@lidi.unlp.edu.ar

Cuando alguien envia un mensaje, el sendmail en primer término preprocesará la dirección haciendo uso de la regla 3 (para la eliminación de cualquier texto que no pertenezca a la dirección en si misma), luego verificará si esa dirección se encuentra en el archivo de alias. De ser así tratará de resolver el o las direcciones y luego hacer el envío efectivo del mensaje, determinando el transporte, máquina destino y usuario, haciendo uso de la regla 0. Evaluemos si alguien tratase de enviar un mensaje a *root* en base al ejemplo anterior. En primer lugar se resolvería enviando dicho mensaje a *arusso@ayelen.fisica.unlp.edu.ar* y a *crusso*, pero el sendmail tratará de verificar que las direcciones obtenidas no sean alias, con lo cual para la primera (sobre nuestro ejemplo) no existe un alias, en consecuencia será enviada a esa dirección pasando a través del transporte seleccionado por la regla 0, con la segunda se encontró un alias que es crusso@lidi.unlp.edu.ar, esta será procesada nuevamente por el sendmail y al no existir un alias para ella será enviada a esa dirección. El sendmail no permite alias recursivos. Para una mayor efectividad en la búsqueda de alias, este lo indexa, para un acceso más eficiente, Ud piense que existen archivos de alias de 3000 líneas, si esos archivos no estuvieran indexados, su búsqueda podría ser lo suficientemente onerosa en tiempo.

Listas de correo electrónico

Las listas de correo electrónico, en el Sendmail, es simplemente un alias donde este haga referencia a más de una dirección efectiva o a otros alias. Trabajemos sobre un ejemplo:

Alias	***Direcciones o alias***
lista_de_usuarios:	*usuarios_rayos_x, usuarios_microondas, usuarios_tecnicos*
usuarios_rayos_x:	*rivero, punte, ellena, piro, echeverria*
usuarios_microondas:	*mamaron, fantoni*
usuarios_tecnicos	*russo, miranda, vina, postmaster*

Esta sería mi organización dentro de la HP-UX 9000/710 utilizando Sendmail como agente de transmisión de mensajes, clasificando a los usuarios en listas según el trabajo o temas de investigación que estos realizan. La lista principal esta compuesta a su ves por sublistas (la de las listas de usuarios_xxxxx). Si enviásemos un mensaje a la lista *lista_de_usuarios* esta se propagaría en las demás listas (usuarios_rayos_x, usuarios_microonadas, usuarios_tecnicos), quien a su ves, estas, podrían propagarse en otras sublistas (en caso de que existieran). De tener una gran cantidad de usuarios, esta la opción de utilizar el comando *include*, este permite la inclusión de archivos que contienen alias o direcciones efectivas. Siguiendo con el ejemplo anterior nos quedaría:

lista_de_usuarios:	*usuarios_rayos_x:, usuarios_microondas, usuarios_tecnicos*
usuarios_rayos_x:	*:include:/usr/sendmail/alias/usuarios_rayos_x*

usuarios_microondas: *:include:/usr/sendmail/alias/usuarios_microondas*
usuarios_tecnicos: *:include:/usr/sendmail/alias/usuarios_tecnicos*
postmaster: *root@ayelen.fisica.unlp.edu.ar*

formato de los archivos de alias:
/usr/sendmail/usuarios_rayos_x
rivero, punte,
ellena, piro,
echeverria

/usr/sendmail/usuarios_microondas
mamaron, fantoni

/usr/sendmail/usuarios_tecnicos
russo
miranda, vina, postmaster

El permitir la posibilidad de incluir archivos, permite al Sendmail otorgar la responsabilidad de la actualización de las listas de usuarios a algún usuario que actuase como postmaster de la lista (por ej: :include: /users/postmaster/ usuarios_rayos_x , observe que el archivos de alias esta en el subdirectorio de los usuarios, en particular del usuario postmaster).

La cola de trabajo (queue)

Una ves que se determinó cual ha de ser el transporte que se ha de utilizar como medio de comunicación, el Sendmail tiene dos opciones, la primera de entrega inmediata del mensaje y la segunda encolarlo para que en un determinado lapso de tiempo (definido por el sistema o por el postmaster) ser procesado por el Sendmail. La forma de encolarlo es generando un archivo con el mensaje en un subdirectorio cuyo nombre será único. Este mecanismo de cola es utilizado en todos los MTA's serios, en particular en los que nos faltan ver (MMDF y PP). Lo que puede variar de un MTA a otro son las formas en que se procesa la cola de trabajo y los motivos por el cual se encolan los mensajes para su posterior procesamiento.

Los motivos del Sendmail son:

a) Si un mensaje no pudo ser entregado satisfactoriamente entonces este es encolado para su posterior reintento.

b) Si el la opción *s* en verdadera entonces el Sendmail encolará todos los mensajes sin excepción, y en caso de haber realizado la transferencia satisfactoriamente removerá el mensaje de la cola.

c) Si el sendmail es ejecutado en modo *queue-only* (la opción -odq) todos los mensajes serán encolados y su transferencia será diferida en un lapso de tiempo definido por el sistema o por el postmaster.

d) Si la sobrecarga del sistema sobrepasa el límite definido entonces se encolará el mensaje para su posterior transferencia.

Es muy común que los MTA's le den la posibilidad al administrador de revisar mediante algún software el estado de la cola de trabajo o el detalle de los mensajes y su estado. El Sendmail no es una excepción, tiene un programa que le permite revisar la cola cuando el administrador lo crea necesario.

Manejo de errores

Son muchos los motivos por el cual un mensaje no ha podido ser entregado satisfactoriamente (no existe la dirección especificada, formato de dirección erróneo, el transporte no funciona, el name server no esta funcionando,...). El Sendmail estándar tiene internamente definido un transporte que sirve para realizar los envíos de mensajes de errores a los usuarios locales o aquellos usuarios remotos cuyos mensajes que han sido recepcionados no han podido ser efectivamente entregados. En el caso de la HP el mensaje es enviado al usuario MAILER-DEAMON quien es el encargado de los errores del Sendmail.

Interacción del Sendmail con el DNS (Domain Name Server)

Cuando el transporte que ha de utilizarse es el TCP/IP, el Sendmail hace uso de la capacidad del NS (Name Server) para obtener información que en definitiva lo llevará a conectarse a través del SMTP con la máquina remota. Para lograrse la comunicación efectiva, debe contarse con el número de IP correspondiente (recuerde que los usuarios hacen uso de los nombres de las máquinas desconociendo en la mayoría de los casos el número de IP) que será obtenido haciendo las consultas necesarias al NS responsable del dominio donde se esta corriendo el Sendmail.

El operador *$[máquina $]* permite consultar el nombre de la máquina al NS quien nos retornará el nombre oficial de ella en un entorno Internet (por ejemplo: $[ayelen $] obtendré como respuesta ayelen.fisica.unlp.edu.ar).

En los NS es muy común ver registros *MX* (Mail eXchanger), estos registros permiten utilizar a máquinas como concentradora del flujo de la correspondencia electrónica. Antes de que el sendmail realice una conexión punto a punto sobre TCP/IP con una máquina remota, primero verificará que no exista un registro MX para esa máquina, de existir utilizará la máquina que se encuentra especificado en el registro MX quien se encargará de entregar la correspondencia a su destino final. Estos registros poseen un campo que indica la prioridad de conexión que esta determinada por un número entero (menor es el número mayor es la prioridad) Trataremos de dar un ejemplo y comentarlo para un mayor entendimiento.

Supongamos tener un registro MX en un NS de la siguiente forma

maquina_a	*IN*	*MX*	*10 maquina_b*
	IN	*MX*	*20 maquina_c*

Si nosotros quisiésemos enviar un mensaje a *usuario*@*maquina_a* sobre el transporte TCP/IP el sendmail al consultar al NS determina que existen dos registros MX para la *maquina_a* con lo cual tratará de conectarse, no directamente sino a través de la máquina_b que es la de mayor prioridad. En caso de no poder hacerlo lo intentará con la *maquina_c* que es la que le sigue en prioridad.

Los registros MX no son recursivos y pueden aplicarse en la definición dentro del DNS caracteres wilcard.

Seguridad

El Sendmail puede ser la entrada a violaciones del sistema si el administrador no prevé con anticipación los llamados agujeros negros. Estos agujeros los podemos clasificar en dos categorías la primera de ellas son las propias del sistema (son aquellas que no pueden ser anticipadas y surgen con el uso, perfeccionándose en las versiones sucesivas, en el caso del Sendmail al ser un sistema muy usado hay un alto porcentaje de que la ultima versión no contenga eso tipos de agujeros negros), y la segunda la que provee el propio administrador al no configurar correctamente al Sendmail. Es muy común que el *uid* del Sendmail sea root, si se lograse violar la seguridad del sendmail al entrar al sistema uno tendría las prioridades del root (los máximos privilegios). Describiremos los puntos que hay que tener en cuenta.

i) La conversación del Sendmail con el SMTP. Puede ser usada para obtener nombre de usuarios o para tratar de entrar al sistema (debug, showq, vrfy, exp).

ii) Controlar los permisos de lectura, escritura y ejecución de archivos alias, y algunos que se utilizan en el *sendmail.cf*

sendmail.cf de la HP-UX 9000/710

El archivo de configuración con el que hemos trabajado en la RISC está expuesto a continuación. Ud podrá ver cada una de las reglas definidas, los transportes, las clases, los derecho de seguridad,... en el archivo que esta a continuación.

```
######################################################################
######################################################################
#####                                                            #####
#####          SENDMAIL CONFIGURATION FILE                       #####
#####                                                            #####
######################################################################
######################################################################
#####  @(#)$Revision: 16.4 $
######################################################################
###                 Localizable Options                            ###
######################################################################
# logging level
OL11
# defer messages to [IPC] mailers if the nameserver is not running
# OI
# delivery mode
Odbackground
# error reporting mode
Oep
# read timeout
Or5m
# queue timeout interval
OT3d
# load average at which low priority messages are queued rather than delivered
Ox8
# load average at which daemon refuses to accept connections
```

```
OX12
# postmaster address which will receive headers of undeliverable messages
OProot@ayelen.fisica.unlp.edu.ar
######################################################################
###                        Other Options                           ###
######################################################################
###                                                                ###
###   HP recommends that these options not be changed.             ###
###                                                                ###
######################################################################
# queue directory
OQ/usr/spool/mqueue
# Save those UN*X From_ lines
Of
# location of alias file
OA/usr/lib/aliases
# temporary file mode
OF0600
# default UID
Ou1
# default GID
Og1
# location of help file
OH/usr/lib/sendmail.hf
# recognize old style as well as new style lists in headers
Oo
# statistics file
OS/usr/lib/sendmail.st
# wait up to 5 minutes for completion of alias db initialization
Oa5
# queue up everything before starting transmission
Os
# send to me, too if in alias expansion
Om
# if the load average exceeds the x option limit, divide the q option
# value by the difference (plus one) between the current load average and
# the x option limit to find the maximum priority value (i.e. minimum
# priority) of messages to send immediately.
Oq10000
# value added to message priority per recipient
Oy1000
# message precedence factor
Oz1800
# value added to message priority per queue run
OZ9000
######################################################################
###  Configuration-Specific Macro and Class Definitions            ###
######################################################################
# site hiding: local sender identified as user@my_site instead of user@my_host
# DY
# class w defines aliases for the local host
```

```
Cw ayelen.fisica.unlp.edu.ar
# macro L defines the "local domain" to which you connect directly for SMTP mail
DLfisica.unlp.edu.ar
# class S defines hosts to which you connect directly for SMTP mail
#FS/etc/hosts.smtp %s
# class U defines known direct UUCP connections
FU|/usr/bin/uuname %s
# UUCP relay for unresolved ! addresses (via UUCP)
DU
# UUCP relay for unresolved ! addresses (via SMTP)
DWunlp.unlp.edu.ar
# SMTP relay for unresolved @ addresses
DSunlp.unlp.edu.ar
# X.400 relay if X.400 delivery agent is not local
DX
# OpenMail relay if OpenMail delivery agent is not local
DZ
# dumb (not RFC 822 compatible) UUCP hosts
CG
# pathalias external nameserver program
DP/usr/bin/uupath

######################################################################
###                  Configuration Version                         ###
######################################################################
DV16.2
######################################################################
###              Required Macro Definitions                        ###
######################################################################
# official domain name of this host for SMTP
Dj$w
# my name
DnMAILER-DAEMON
# UNIX header format
DlFrom $g $d
# delimiter (operator) characters
Do.:%@!^=/[ ]|
# format of a total name
Dq$?x$x <$g> $|$g$.
# SMTP banner
De$j HP Sendmail ($v/$V) ready at $b
######################################################################
###                  Message Precedences                           ###
######################################################################
Pfirst-class=0
Pspecial-delivery=100
Pjunk=-100
######################################################################
###                    Trusted Users                               ###
######################################################################
Troot
```

```
Tdaemon
Tuucp
Tx400
###################################################################
###              Header Field Formats                           ###
###################################################################
HReceived: $?sfrom $s $.by $w$?r with $r$.
    ($v/$V) id $i; $b
HResent-Date: $a
HDate: $a
HResent-From: $q
H?F?From: $q
H?x?Full-Name: $x
H?P?Return-Path: <$g>
HSubject:
# HPosted-Date: $a
# HReceived-Date: $b
# HResent-Message-Id: <$t.$i@$w>
# H?M?Message-Id: <$t.$i@$w>
###################################################################
###################################################################
#####                                                         #####
#####         Address Rewriting Rulesets                      #####
#####                                                         #####
###################################################################
###################################################################
###################################################################
###  Ruleset 1 - Sender Field Pre-rewriting                     ###
###################################################################
S1
R$+                     $:$>6$1                 strip my_host and canonicalize
R$*<@$+>$*              $@$1<@$2>$3             already has (remote) domain
R$+/$*/$*/$*/$*         $:$1/$2/$3/$4/$5<@$w>   @my_domain on local X.400
sender
###################################################################
###  Ruleset 2 - Recipient Field Pre-rewriting                  ###
###################################################################
S2
R$+                     $:$>6$1                 strip my_host and canonicalize
R$*<@$+>$*              $@$1<@$2>$3             already has (remote) domain
R$+/$*/$*/$*/$*         $:$1/$2/$3/$4/$5<@$w>   @my_domain on local X.400
recpt
###################################################################
###  Ruleset 3 - Address Internalization                        ###
###################################################################
S3
# handle "From:<>" special case
R<>                     $@$n                    null address =>
MAILER-DAEMON
# basic textual canonicalization
R$*<$*<$+>$*>$*   $1<$2$3$4>$5                  strip <> from inside
```

```
R$*<$+>$*          $2                    strip phrase and < >
R$+ at $+          $1@$2                 RFC 733 at => RFC 822 @
# source route <@a,@b,@c:user@d> syntax to internal form <@a>:@b:@c:user@d
R@$+,$+            @$1:$2                change all , to :
R@$+.UUCP:$+       $@<@$1.UUX>:$2        .UUCP pseudo-domain in route
R@$+:$+            $@<@$1>:$2            focus on next hop
# The @ delimiter takes precedence.  Leave this alone.
R$+@$+             $:$1<@$2>             focus on domain
R$+<$+@$+>         $1$2<@$3>             move gaze right
R$+<@$+.UUCP>      $@$1<@$2.UUX>         .UUCP pseudo-domain
R$+<@$+>           $@$1<@$2>             already in internal form
# The ! delimiter.
R$+^$+             $1!$2                 convert obsolete ^ to !
R$+!$+             $@$2<@$1.UUX>         host!user =>
user<@host.UUX>
# % is a low precedence @.
R$+%$+             $:$1<%$2>             focus on domain
R$+<$+%$+>         $1$2<%$3>             move gaze right
R$+<%$+>           $1<@$2>               user%host => user@host
R$+<@$+.UUCP>      $@$1<@$2.UUX>         .UUCP pseudo-domain
# miscellaneous cleanup
R$+@               $@$1                  user@ => user
R$+%               $@$1                  user% => user
###########################################################
###  Ruleset 4 - Final Output Post-rewriting                ###
###########################################################
S4
# special cases
R$+<@>             $1                    null domain
# UUCP must always be presented as host!user
R$+<@$+.UUX>       $@$2!$1               user<@host.UUX> =>
host!user
# UUCP hop in source route
R<@$+.UUX>:$+      <@$1.UUCP>:$2         .UUX in source route =>
.UUCP
R<@$+>:$+.UUX$+    <@$1>:$2.UUCP$3       .UUX in source route =>
.UUCP
# defocus
R$*<$+>$*          $1$2$3                remove internal form < >
# don't change %s or @s in mixed addresses
R$+!$+@$+          $@$1!$2@$3            don't interpret it any further
R$+!$+%$+          $@$1!$2%$3            don't interpret it any further
# restore source route to external form
R@$+:$+:$+         @$1,$2:$3             all but last : => ,
R@$+               $@<@$1>               add < > to protect the ,s
# should be exactly one @ in user@domain style address
R$+%$+             $1@$2                 all % => @
R$+@$+@$+          $1%$2@$3              all but last @ => %

###########################################################
###  Ruleset 0 - {Delivery_Agent, Host, User} Resolution    ###
```

```
##########################################################
S0
# recognize local host or canonicalize
R$+                     $:$>6$1                 anything to ruleset 6 once
# resolve domain-literals (numeric internet addresses) not canonicalized above
R$*<@[$+]>              $#tcp$@[$2]$:$1@[$2]    user@internet address
R<@[$+]>:$*             $#tcp$@[$1]$:@[$1]:$2   internet address in source route
# resolve mail to dumb UUCP hosts
R$+<@$=G.UUX>           $#dumbuucp$@$2$:$1      user@dumb_host.UUX
# resolve mail to other known UUCP hosts
R$+<@$=U.UUX>           $#uucp$@$2$:$1          user@host.UUX
R<@$=U.UUX>:$+          $#uucp$@$1$:$2          @host.UUX in source route
# try to get a path to an unresolved UUCP address from pathalias nameserver
# R$+<@$-.UUX>          $:$>5$2!$1              uupath pathalias routing
# R$+<@$=G.UUX>         $#dumbuucp$@$2$:$1      to dumb UUCP host
# R$+<@$=U.UUX>         $#uucp$@$2$:$1          to other known UUCP host
# pass unresolved UUCP addresses to the UUCP relay (via SMTP)
R$+<@$+.UUX>            $#tcp$@$W$:$1<@$2.UUX>      to UUCP relay via SMTP
# pass unresolved UUCP addresses to the UUCP relay (via UUCP)
# R$+<@$+.UUX>          $:$>3 $U!$2!$1          re-internalize with $U
# R$+<@$=G.UUX>         $#dumbuucp$@$2$:$1      to dumb UUCP relay
# R$+<@$+.UUX>          $#uucp$@$2$:$1          to UUCP relay
# other UUCP addresses are in error
R$*<@$+.UUX>$*          $#error$:unable to route to UUCP host name $2
# select hosts to connect with directly for SMTP mail:
# connect to hosts in class S
R$+<@$=S>               $#tcp$@$2$:$1<@$2>      user@domain
# connect to hosts in local domain
# R$+<@$+.$L>           $#tcp$@$2.$L$:$1<@$2.$L>    user@host.localdomain
# try to connect to any host for user@domain
# R$+<@$+>              $#tcp$@$2$:$1<@$2>      user@domain
# try to connect to any host for source route
R<@$+>:$+               $#tcp$@$1$:<@$1>:$2     source route
# pass unresolved SMTP addresses to the SMTP relay (don't relay source routes)
R$+<@$+>                $#tcp$@$S$:$1<@$2>      user@domain to SMTP relay
# other SMTP addresses are in error
R$*<@$+>$*              $#error$:unable to route to domain $2
# file names, programs, and :include: must resolve to local mailer;
# explicitly distinguish these from X.400 and OpenMail syntax
R/$*                    $#local$:/$1            to absolute file path name
R|$*                    $#local$:|$1            to a program
R:include:$*            $#local$::include:$1    to :include: list
# resolve X.400 mail: local host is X.400 gateway
# R$+/$*/$*/$*/$*       $#x400$@$w$:$1/$2/$3/$4/$5
# resolve X.400 mail: remote host is X.400 gateway
# R$+/$*/$*/$*/$*       $#tcp$@$X$:$1/$2/$3/$4/$5<@$X>
# resolve mail to OpenMail: local host is OpenMail gateway
# R$+/$*                $#openmail$@$w$:$1/$2
# resolve mail to OpenMail: remote host is OpenMail gateway
# R$+/$*                $#tcp$@$Z$:$1/$2<@$Z>
# by default, reject X.400 address as error
```

```
R$+/$*/$*/$*/$*        $#error$:X\.400 delivery agent not configured
# by default, reject OpenMail address as error
R$+/$*                  $#error$:OpenMail delivery agent not configured
# resolve mail to OpenMail from remote OpenMail system
# Ropenmail             $#omxport$@$w$:openmail
# resolve mail to X.400 from OpenMail
# Ropenmailx4           $#omx400$@$w$:openmailx4
# other addresses must resolve to local mailer in order for mail to Full.Name,
# command line aliases, and quoted user names (\user) to be delivered.
# remaining names must be local
R$+                     $#local$:$1             name
######################################################################
######################################################################
#####                                                            #####
#####          Special Rulesets                                  #####
#####                                                            #####
######################################################################
######################################################################
######################################################################
### Ruleset 5 -   Pathalias Nameserver                             ###
######################################################################
S5
R$+             $:$<P$1                 uupath pathalias routing
R$+             $:$>3$1                 re-internalize
######################################################################
### Ruleset 6 -  Local Host Recognition                            ###
######################################################################
S6
# RFC 822 does not permit hostnames to end in .
R$*<@$*.>$*             $1<@$2>$3               strip trailing .
# strip local host
R$+<@$w>                $>3$1                   strip my_host and re-internalize
R<@$w>:$+               $>3$1                   strip my_host and re-internalize
# recognize local host in UUCP syntax
R$+<@$k.UUX>            $:$1<@$w>               my_host in UUCP syntax
# Recognize mail from uucp for x400 user on this system
# R$+<@hpx400.UUX>      $:$1<@$w>               X.400 in UUCP syntax
R<@$k.UUX>:$+           $:<@$w>:$1              my_host.UUCP in source route
R$*<@$+.UUX>$*          $@$1<@$2.UUX>$3         don't canonicalize host.UUX
# canonicalize host and possibly recurse
R$*<@$+>$*              $:$1<@$[$2$]>$3         user@host or source route
R$*<@$=w>$*             $:$>6$1<@$w>$3          recurse if still to my_host
######################################################################
######################################################################
#####                                                            #####
#####     Mailer (Delivery Agent) Definitions                    #####
#####                                                            #####
######################################################################
######################################################################
######################################################################
###          local and program mailers                           ###
```

```
#############################################################
Mlocal,      P=/bin/rmail, F=DFMPlms,  S=10,  R=20,  A=rmail -d $u
Mprog,       P=/bin/sh,  F=DFMPlshu,       S=10,          R=20,  A=sh -c $u
S10
R$*<@$+.UUX>                    $@$1<@$2.UUX>   don't modify UUCP address
R$*<@$+>$*          $@$1<@$2>$3             already has domain
R$+                 $:$1<@$w>               add local domain to user
S20
R$*<@$+.UUX>        $@$1<@$2.UUX>           don't modify UUCP address
R$*<@$+>$*          $@$1<@$2>$3             already has domain
R$+                 $:$1<@$w>               add local domain to user
#############################################################
###             SMTP TCP/IP mailer                                ###
#############################################################
Mtcp,  P=[IPC],     F=CDFMXmu,  S=11,  R=21,  E=\r\n,  A=IPC $h
S11
R$*<@$+.UUX>        $@$2!$1<@$w>            add local domain to UUCP
address
R<@$+>:$*           $@<@$w>:@$1:$2          add local domain to source
route
R$+<@$+>            $@$1<@$2>               already has domain
R$+                 $:$1<@$?Y$Y$|$w$.>      add local domain
S21
R$*<@$+>$*          $@$1<@$2>$3             already has domain
R$+                 $:$1<@$w>               add local domain
#############################################################
###             UUCP mailer                                       ###
#############################################################
Muucp,      P=/usr/bin/uux,         F=DFMUshu,  S=13,  R=23,  A=uux - $h!rmail ($u)
S13
R$+<@$+.UUX>        $@$2!$1<@$k.UUX>        host!user => my_host!host!user
R$+@$+<@$+>         $1%$2<@$3>              all but last @ => %
# enable uucp recipient to reply to remote x400 sender
# R$*/$*<@$+>       $@hpx400!$1/$2%$3<@$k.UUX>  user@host => my_host!hpx400!user%host
R$+<@$+>            $@$1%$2<@$k.UUX>        user@host => my_host!user%host
R<@$+>:$+     $@<@$k.UUCP>:@$1:$2    prepend @my_host.UUCP to route
# enable uucp recipient to reply to local x400 sender
# R$*/$*            $@hpx400!$1/$2<@$k.UUX>         user => my_host!hpx400!user
R$+                 $@$1<@$k.UUX>           user => my_host!user
S23
#############################################################
###             Dumb UUCP mailer                                  ###
#############################################################
###     UUCP for hosts running non-RFC 822 mailers                ###
#############################################################
Mdumbuucp,  P=/usr/bin/uux,  F=DMUshux,      R=23,          A=uux - $h!rmail ($u)
#############################################################
```

```
###                X.400 mailer                                    ###
#################################################################
Mx400, P=/usr/lib/x400/x4mailer, F=CDMFmn, S=14, R=24, A=x4mailer -f $g $u
S14
S24
#################################################################
###                OpenMail mailers                                ###
#################################################################
Mopenmail, P=/usr/openmail/bin/unix.in, F=DFLMXmnu, S=15, R=25, A=unix.in
S15
S25
Momxport, P=/usr/openmail/bin/xport.in, F=LMn, A=xport.in $u
Momx400, P=/usr/openmail/bin/x400.out, F=LMn, A=x400.out $u
```

Multichannel Memorandum Distribution Facility (MMDF)

Introducción

Como en el Sendmail, el MMDF también cuenta con un archivo de configuración, con la misma finalidad que el sendmail.cf. En este caso este archivo se llama *mmdftailor*. Como veremos a continuación la sintaxis utilizada, difiere completamente a la empleada en el sendmail.cf. Entran en juego otras variables en función de la filosofía de implementación adoptada. En él se definen todos los atributos de la máquina local, tales como, el nombre, el dominio al que pertenece, como configurar los canales, los alias.... El MMDF utiliza otro criterio de selección de los trasportes para realizar la entrega efectiva del mensaje. Es por eso que hemos seleccionado este MTA, no sólo por ser también un estándar como el Sendmail, sino también por el enfoque que le dieron sus creadores. El MMDF es un producto software con un conjunto de programas donde el más importante es el *deliver*, y el *submit* estos son los encargados de enviar y escuchar a los diferentes transportes por la entrada y salida de mensajes. Aquí no vamos a hablar de transportes, sino que introduciremos el concepto de canal (este concepto será comentado en breve).

Los ejemplos que daremos a medida que vayamos desarrollando la explicación de este MTA, se basan en una configuración concreta sobre la red de la UNLP (en el Dto. de Física) en una máquina cuyo nombre es *venus.fisica.unlp.edu.ar.* La red de la UNLP esta en pleno desarrollo, con lo cual esta máquina cuenta con la característica de tener que mantener cierta compatibilidad del direccionamiento anterior (conexión UUCP) con el nuevo (direcciones Internet) a medida que los usuarios vayan migrando de un formato de direcciones a otro.

Sintaxis del archivo de configuración

En el mmdftailor no existen variables propiamente dichas ni actúan como tal (referenciando su valor como en el caso del Sendmail), estos son llamadas *palabras claves* (por ejemplo: palabra_clave parámetro1, parámetro2, parámetro3,...). En algunos casos estas palabras claves pueden actuar como variables (por ejemplo: palabra_clave parámetro1). A continuación mostraremos en una tabla las palabras claves que intervienen en el mmdftailor con una breve descripción de sus funciones, para luego incursionar más detenidamente en cada una de ellas, comentando las funcionalidades de los posibles parámetros que pueden tener.

Palabra Clave	***Significado***
MLDOMAIN	Aquí se define el dominio superior al cual pertenece la máquina que será en definitiva la que ejecute el programa
MLNAME	Se especifica el nombre de la máquina o el nombre del subdominio al que pertenece en caso de ocultar el nombre de la máquina
MLOCMACHINE	Se especifica el nombre de la máquina para el ocultamiento de ella

UUname	Se especifica el nombre de la máquina para conexiones con el canal UUCP
MSUPPORT	Setea la dirección del postmaster para enviarle los mensajes que no puedan ser entregados.
MTBL	Asocia un nombre abreviado con un nombre más descriptivo para los archivos de los alias, dominios y canales.
ALIAS	Define los posibles orígenes de la información de los alias haciendo uso de los nombres especificados en MTBL
MCHN	Define los canales disponibles con el que cuenta MMDF para el transporte
MDMN	Describe los dominios conocidos para el MMDF
MMSGLOG	Registra como nivel de seguridad las llamadas a los programas deliver y submit
MCHANLOG	Controla las entradas que el MMSGLOG no controla
MLCKTYPE	Especifica el tipo de protocolo utilizado para el mailbox

Las palabras claves MTBL, ALIAS, MDMN y MCHN interactuan juntas. Estas intervienen en la configuración de los alias, dominios y canales. Se los introducirá a medida que hablemos sobre cada uno de esos temas.

MLDOMAIN

La palabra clave MLDOMAIN describe el dominio superior o subdominio usado por la máquina local. Veamos un ejemplo:

Nombre de la máquina dentro de Internet venus.fisica.unlp.edu.ar
máquina venus
dominio fisica.unlp.edu.ar

***MLDOMAIN** fisica.unlp.edu.ar*

MLNAME y MLOCMACHINE

Generalmente el MLNAME describe el nombre de la máquina local. Siguiendo con el ejemplo anterior nos quedaría

***MLNAME** venus*

Por lo general el correo de un subdominio es controlado por una máquina específica, entonces si se hace referencia a esa máquina, cuando esa máquina quede fuera de servicio por alguna razón ajena a los usuarios, se tendría que cambiar las direcciones de ellos a una nueva dirección que sirva de soporte ante estos inconvenientes. Pero el cambiar de dirección es más drástico de lo que parece. Existen usuarios que se encuentran subscritos a listas con lo cual no es nada trivial hacer los cambios mas aún si eso es tan solo por algunos días. La solución a ese

inconveniente es ocultar el nombre de la máquina y cuando algún error ocurriese se cambia la máquina, pero no altera en nada las direcciones. Si se desea ocultar el nombre de la máquina atrás del dominio, el MLNAME describe el subdominio al que pertenece y el MLOCMACHINE pasará a describir el nombre de la máquina. Siguiendo con el mismo ejemplo nos quedaría:

MLNAME *fisica*
MLOCMACHINE *venus*

UUname

Este describe el nombre usado para una conexión en UUCP. En nuestro caso dentro del Dto. de Física el nombre de UUCP no coincide con el de Internet. En nuestra configuración el nombre de UUCP es fisilp, quedándonos:

UUname *fisilsp*

MSUPPORT

Describe la dirección del administrador del sistema de correo electrónico. Sobre Internet Ud. puede definir al administrador como postmaster.

MSUPPORT *root@ayelen.fisica.unlp.edu.ar*

Manejo de alias

Los alias son manejados de forma distinta que el Sendmail. En primer lugar se le debe decir al MMDF cuales son los archivos que intervienen, para eso, se utiliza la palabra clave MTBL. En el MTBL tiene definido los parámetros *name, file* y *show*. El name le asigna un seudonombre a la tabla definida en file, que será usada cuando se le quiera hacer referencia en el mmdftailor. El parámetro show sirve para que en los programas de seguimiento o debug aparezca como comentario cuando se acceda a estas tablas.

MTBL name = alias, file = "alias.ali", show = "Administrative aliases"
MTBL name = lalias, file = "alias.list", show = "Mailing list aliases"
MTBL name = auser, file = "alias.user", show = "General user aliases"

La palabra clave ALIAS define las tablas que intervendrán como alias. Los parámetros que intervienen son *table, nobypass* y *trusted*. El parámetro table enlaza el archivo referenciado en el MTBL como archivo de alias. El parámetro nobypass no permite que las direcciones de los usuarios locales sean salteadas por el mecanismo de trabajo sobre el archivo especificado en file y el parámetro tursted permite que en los alias se puedan especificar archivos o pipes. Su modo de empleo es el siguiente.

ALIAS table = alias, nobypass, trusted
ALIAS table = lalias, nobypass
ALIAS table = auser

Las tres definiciones anteriores hacen referencia a las listas correo (lalias), a los alias de los usuarios (auser) y a los alias que son usados para el manejo administrativo del MMDF (alias). La estructura interna de los archivos de los alias de los usuarios y de los alias de administración es la siguiente:

Nombre del alias	***valor o valores***
adrian:	*arusso@ayelen.fisica.unlp.edu.ar*

Cuando alguien envia un mensaje a un usuario el submit tratará de procesar la dirección. En primer término el MMDF seleccionara las tablas de alias definidas por el administrador para tratar de resolver la dirección dada buscando en ellas alguna referencia que machee con la dada (por ejemplo: si alguien enviase un mensaje a *adrian* este indirectamente será enviado a *arusso@ayelen.fisica.unlp.edu.ar*). Existen alias que cumplen funciones administrativas (postmaster, root,...), el MMDF diferencia estas direcciones de las tradicionales especificándolas en forma separada (véase como se lo definió en el ejemplo anterior). Es probable que el administrador del sistema no sea necesariamente el usuario root, podría ser que sea un usuario remoto, en ese caso el alias postmaster contendrá la dirección o alias del administrador remoto. Veamos un ejemplo:

Tabla alias
postmaster: claudia

Tabla auser
claudia: crusso@lidi.unlp.edu.ar

Resultado
mail postmaster ----> mail crusso@lidi.unlp.edu.ar

Manejo de Listas

En el archivo de alias para listas de correo definidas en el ejemplo anterior se introducen explícitamente el concepto de *administrador de lista de correo*. Si bien en el Sendmail se había introducido ese concepto implícitamente, es aquí donde se le da a las listas de correo y a sus administradores un formalismo coherente y apropiado a las necesidades de los usuarios. En los archivos de listas de correo se definen, el nombre, el contenido y el administrador de la lista. Veamos un ejemplo: supongamos que deseamos crear una lista con los usuarios de ayelen.fisica.unlp.edu.ar, la forma de especificarla es la siguiente:

Tabla lalias

profimo:	*profimo-outbound@list-processor*	*#nombre de lista*
profimo-outbound:	*maranon,fantoni,punte,rivero,ellena,piro,gustavo,russo*	*#integrantes*
profimo-request:	*russo@ayelen.fisica.unlp.edu.ar*	*#administrador*

Las palabras claves *outbound* y *request* acompañadas con el nombre de la lista (por ejemplo: profimo) definen los integrantes y la dirección del administrador. La dirección *list-processor* es una palabra reservada del MMDF que sirve para que los mensajes sean procesados por un canal específicamente diseñado a tal fin. La

característica del *include* ya comentada en el Sendmail es también adoptada por el MMDF.

profimo: profimo-outbound@list-processor #nombre de lista
profimo-outbound: ":include:/tmp/lista_de_usuarios_del_profimo"
#integrantes
profimo-request: russo@ayelen.fisica.unlp.edu.ar
#administrador

Manejo de dominios

MMDF hace uso de las tablas de los dominios que sirven para dos propósitos, el primero para informar al MMDF como las máquinas estan interconectadas y la segunda para especificar las consideraciones de ruteo. El manejo de dominio propiamente dicho esta definido en archivos al igual que las alias. Los archivos de dominios para otros dominios que no sea el local se lo nombra con el nombre del dominio y la extensión *.dom* (por ejemplo: si crease un archivo de dominios para el dominio *edu.ar* el archivo sería *edu.dom*). Existen cuatro archivos de dominios por omisión que son:

dominio	***nombre del archivo***	***Descripción***
local	*local.dom*	*máquina local*
domain	*domain.dom*	*máquinas en el dominio local*
uucp	*uucp.dom*	*máquinas en el dominio UUCP*
root	*root.dom*	*dominios no listados en otros archivo de dominio*

La definición de un dominio comienza con la palabra clave MDMM, utilizando como parámetros a *name, show, dmn* y *table*. El name contiene una abreviatura del dominio, dmn contiene el nombre completo del dominio, show contiene una línea de caracteres que es la mostrada cuando se ejecutan programas de seguimiento o debug y table es el nombre de la tabla asociada al dominio, definida en MTBL. El ejemplo que está a continuación nos muestra el enlace entre la tabla de dominios con su especificación (MDMN).

MTBL name = smtpdom, flags = ns, show = "Local Ethernet", flags = domain, flags = partial
MTBL name = locdom, file = "local.dom", show = "Local Domain", flags = dbm
MDMN name = "fisica", dmn = "fisica.unlp.edu.ar", show = "Local domain", table = locdom

Observe que existen algunas diferencias con la anterior definición del MTBL. Se introducen los parámetros *flags = valor*. Cuando flags tiene el valor ns, se le indica en ese caso que tendrá que consultar al name server para evacuar cualquier dato referente a alguna máquina. El flags = domain esta indicando que busque una dirección dada en el dominio especificado en el parámetro dmn en la definición del dominio. El formato de los archivos de dominios es el siguiente: para el local.dom que describe la máquina local que es venus dentro del dominio física.unlp.edu.ar nos quedaría:

máquina	*máquina + dominio*
venus:	*venus.fisica.unlp.edu.ar*

Canales

MMDF introduce el concepto de *canal*. El canal es simplemente un programa compilado que le permite a la máquina hablar con los diferentes protocolos de comunicación. El deliver y el submit los ve como puerta de acceso a los diferentes transportes (UUCP, TCP-IP,...). Generalizando esta idea el MMDF incorpora otros canales (badhost, baduser, list) que precisamente no son transportes que actúan sobre un determinado protocolo (TCP-IP, UUCP, X400) sino que reciben mensajes para un futuro procesamiento (este tipo de canales permite al MMDF interactuar con el sistema operativo), pero que en definitiva son configurados de la misma manera que los transportes. Esta característica tiene la ventaja de modularizar los problemas de transportes y manejo del sistema operativo en ellos, sin tener que modificar en principio el deliver. Los diferentes canales son:

Nombre del canal	Función
smtp	Este programa llama al deliver para la entrega o arribo de mensajes desde TCP/IP. El canal smtp transfiere mensajes estableciendo una conexión con una máquina remota usando el SMTP para enviar uno o más mensajes, permite el intercambio de mensajes con cualquier máquina en la red de Internet.
uucp	Este canal recibe o envia mensajes a máquinas remotas haciendo uso del protocolo UUCP. En caso de recibir mensajes por este protocolo los transferirá al deliver quien se encargará de su entrega final a usuarios locales o lo ruteará a otros MTA's.
local	Es llamado para entregar mensajes a cada uno de los mailbox de los usuarios locales
badhost	Este canal es llamado cuando no es conocida una máquina específica. Cualquier mensaje a máquinas desconocidas es enviado a este canal
baduser	Este canal cumple la misma función que el badhost, con la diferencia que se trata de usuarios desconocidos dentro del sistema.
list	Este canal es utilizado para las listas de usuarios. La ventaja de su uso es que trata de mejorar la performance del sistema cuando se envia listas extensas.
delay	Este canal es utilizado cuando se consulta al Name Server.

Para cada canal el MMDF provee dos programas, uno para la entrada de los mensajes y otro para la salida. Los programas que reciben mensajes son llamados server a diferencia de los que envian que son llamados clientes. La definición del canal en el archivo de configuración se hace utilizando la palabra clave MCHN, veamos el siguiente ejemplo:

MTBL *name = local, file = "local.chn", show = "Local Host Aliases"*
MTBL *name = locdom, file = "local.dom", show = "Local Domain"*
MCHN *name = local, show = "Local Delivery", ap = 822, tbl = local, mod = imm*
MDMN *name = "fisica.unlp.edu.ar", dmn = "fisica.unlp.edu.ar", show = "Local domain", table = locdom*

Los parámetros ap = 822 significa la regla de parsing asociada a la dirección, en esa caso se parseará en formato RFC-822, el parámetro mod = imm significa que el canal será invocado inmediatamente y el tbl = local asocia el canal con la tabla local definida en MTBL correspondiente. El archivo asociado al canal contiene los host con los cuales se conectará a través de ese canal.

El canal badhost que le permite al administrador redireccionar todos aquellos mensajes que no sean manejados por el MTA local (serviría como Mail-Relay).

Como el MMDF rutea los mensajes

Los mensajes arriban haciendo uso de los canales definidos para tal fin. El MMDF submit acepta los mensajes de los canales de entrada y determina el correcto canal de salida haciendo uso de la dirección de destino. El submit utiliza la información de los dominios especificada en el archivo de configuración para reconocer la máquina de la dirección de destino. Basándose en la descripción de la máquina destino el submit seleccionará el canal correspondiente para luego introducir el mensaje en la cola correspondiente a dicho canal.

Algoritmo de búsqueda sobre la tabla de dominios

Cuando al submit le llega un mensaje, este toma la dirección de destino y la descompone en máquina + dominio y trata de ubicar la máquina en las tablas de dominio. Por ejemplo si la dirección fuera *root@ayelen.fisica.unlp.edu.ar*, el nombre del dominio sería *fisica.unlp.edu.ar* y la máquina sería *ayelen*, el nombre de la tabla a buscar será *fisica.unlp.edu.ar* y la máquina a buscar dentro de esa tabla será *ayelen*. El MMDF testea una dirección macheando contra los nombres de dominios definidos en las entradas MDMN. Por ejemplo: *MDMN name = "fisica.unlp.edu.ar", show = "Dto. de Física", tbl = fisica* entonces el MMDF buscaría la máquina en archivo definido en el MTBL con el nombre *fisica*, *MTBL name = fisica, file = "fisica.dom", show = "Fisica Delivery"* de encontrarse se tratará de ubicar al canal correcto haciendo uso de el nombre obtenido en la tabla de dominio. De no encontrarse entonces se accederá al dominio root.

La cola de trabajo (queue)

El MMDF tiene tantas colas de trabajo como canales posea, es decir por ejemplo que los mensajes que sean transferidos por el canal smtp serán ubicados en una cola de trabajo diferente a si es procesado por el canal uucp. Esta clasificación es más organizativa que la del Sendmail. Recuerde que el Sendmail posee una única cola de trabajo para todos los mensajes. Cada uno de los mensajes tendrá una identificación única dentro de la cola de trabajo.

El MMDF al igual que el Sendmail ofrece programas para controlar la colas de trabajo.

Interacción del MMDF con el DNS (Domain Name Server)

La interacción del MMDF con el DNS, lo hace a través de las tablas de dominios. Cuando a una tabla especificada con la palabra clave MTBL se le define el parámetro *ns* se le esta diciendo que acceda al NS para obtener todos los datos referente a la máquina destino. Observe que en el ejemplo que está a continuación las tablas corresponden a tablas que son utilizadas para el dominio y para el canal smtp. También al igual que el Sendmail hace uso de los registros MX del name server.

```
MTBL    smtpchn, flags=ns, show="SCO SMTP Channel", flags=channel, flags=partial
MTBL    smtpdom, flags=ns, show="Local Ethernet", flags=domain, flags=partial
```

Seguridad

Tiene un alto nivel de seguridad, aprendiendo de los errores del Sendmail, el MMDF corre como usuario mmdf, lo que garantiza en cierta manera que los derechos del MTA no sean los privilegiados (los de root).Si alguien violase el sistema de seguridad este podrá acceder unicamente a aquellos archivos donde tenga permiso el MMDF. El daño ocurriría sobre los mensajes pero el sistema en su totalidad quedaría íntegramente protegido.

mmdftailor de venus.fisica.unlp.edu.ar

A continuación se encuentra el archivo de configuración utilizado en la máquina venus.fisica.unlp.edu.ar que actualmente esta ofreciendo el servicio. Su puesta en funcionamiento data de Diciembre de 1993 preparada para una futura conexión Internet.

```
;
;   mmdftailor (Thu, 30 Sep 93 15:49:45)
MLDOMAIN unlp.edu.ar
MLNAME fisica
MLOCMACHINE venus
UUname fisilp
MSUPPORT root@ayelen.fisica.unlp.edu.ar
; Alias configuration
MTBL    name=alias, file="alias.ali",   show="Administrative aliases"
MTBL    name=lalias, file="alias.list",         show="Mailing list aliases"
MTBL    name=auser, file="alias.user", show="General user aliases"
ALIAS   table=alias, nobypass, trusted
ALIAS   table=lalias, nobypass
ALIAS   table=auser
; Local mail configuration info
MTBL    name=local, file="local.chn", show="Local Host Aliases"
MTBL    name=locdom,        file="local.dom", show="Local Domain"
MCHN    local, show="Local Delivery", ap=822, mod=imm
MDMN    "fisica.unlp.edu.ar", show="Local domain", table=locdom
; special list processing configuration info
MTBL    list,  file="list.chn",  show="List Channel"
MCHN    list, show="List Processing", ap=same, mod=imm
; SMTP Configuration info
```

```
; The local domain table
MTBL    smtpchn, flags=ns, show="SCO SMTP Channel", flags=channel,
flags=partial
MTBL    smtpdom, flags=ns, show="Local Ethernet", flags=domain, flags=partial
MCHN    smtp, show="SCO SMTP Delivery", ap=822, tbl=smtpchn, mod=imm,
        confstr="venus.fisica.unlp.edu.ar", pgm=smtp
MCHN    delay, show="Name server Delay Channel", ap=same, tbl=smtpchn
MDMN    "fisica.unlp.edu.ar", show="Local Ethernet", table=smtpdom
; UUCP Configuration info
MTBL    uuchn, file="uucp.chn", show="SCO UUCP Channel"
MTBL    uudom, file="uucp.dom", show="SCO UUCP Domain", flags=route,
flags=partial
MCHN    uucp, show="SCO UUCP Delivery", tbl=uuchn, ap=same, mod=imm
MDMN    "UUCP", show="UUCP Domain", table=uudom
; The badhosts channel
MCHN    badhosts, show="Last-Chance Routing", pgm=smtp, tbl=smtpchn,
ap=822,
        mod=imm, host="unlp.unlp.edu.ar"
; The baduser channel
MCHN    badusers,show="Last-Chance Routing", pgm=uucp, tbl=uuchn, ap=822,
        mod=imm,   host="secyt.uucp"
; The root domain table.
MTBL    rootdom, file="root.dom", show="Root Domain", flags=route
MDMN    "", show="Root Domain", table=rootdom
;  Logging levels
MMSGLOG         level=FAT, size=20
MCHANLOG        level=FAT, size=20
MLCKTYPE        advisory
```

Postman Pat (PP)

Introducción

En Agosto 1988 S.E.Kille presentó en la Conferencia de Aplicaciones Distribuidas y Manejo de Sistemas de Mensajes un paper sobre el diseño e implementación de un nuevo MTA, llamado PP. S.E.Kille representaba a el Departamento de Ciencias de la Computación de la Universidad de Londres. Hoy este MTA, fundamentalmente en Europa está muy difundido. Nosotros hemos traído los fuentes de la versión 6.0 de PP de Australia, que es una versión Alfa. La versión 6.5 esta en pleno desarrollo y existen en la actualidad versiones Beta ejecutándose como banco de pruebas en diferentes lugares de Europa y los EEUU. Se compiló el programa en una HP-UX 9000/710. Actualmente el PP se encuentra funcionando en dicha computadora. El tiempo de puesta apunto de este producto, nos llevó aproximadamente de cuatro a cinco meses, fundamentalmente porque PP interactúa con las librerías de ISODE, cuyos fuentes tuvieron que ser traídos desde los EEUU por ftp. El PP más el ISODE son aproximadamente 13Mb de fuentes en lenguaje C los cuales sufrieron modificaciones en algunos puntos para adaptarlo a las características de la máquina en la cual se estaba ejecutando.

El PP al igual que sus antecesores (Sendmail y MMDF) también tiene un archivo de configuración llamado *tailor*. Ese archivo tiene la misma funcionalidad que el sendmail.cf y el mmdftailor. Su filosofía conceptual esta orientada a la implementada en el MMDF. Mantiene el concepto de canales. El control y manejo de alias y dominios se realiza con tablas indexadadas al igual que MMDF, lo que varía es el flujo y control que serán evaluados en los próximos párrafos.

Los programas más importantes que componen a PP son dos, el primero es el *submit* que es invocado por los canales de entrada cuando arriban los mensajes y el segundo el manejador de la cola de trabajo llamado *QMGR* que se encarga de su administración y también de llamar a los canales de salida para la entrega o ruteo de los mensajes.

Canales

PP ofrece la posibilidad de procesar por separado las direcciones y los contenidos de los mensajes. Es una particularidad muy importante a la hora de migrar de un formato estándar como lo es el RFC-822 a otro como el X400(84). Este manejo se realiza a través de filtros (los filtros estan especificados en forma estándar en RFC's como así también en recomendaciones publicadas por la CCITT) implementados por PP, y definidos en los canales que se encuentran en el archivo de configuración. Los canales se pueden ver como entidades que reciben un mensaje como entrada y generan un mensaje como salida. Cuando se migra de un formato a otro se recibe un mensaje, por ejemplo en RFC-822 y pasando por un canal de filtro podría retornar un mensaje en formato X400(84).

Los canales estan agrupados en dos, los canales de mandato y los canales opcionales. Los canales de mandato son los encargados de las acciones de control y mantenimiento de la cola de trabajo. Los canales opcionales son aquellos canales que actúan sobre los protocolos de comunicación (TCP/IP, UUCP, DECNet,...) y además contienen los filtros que permiten migrar direcciones y cuerpos de los mensajes de un formato a otro. Las tablas que estan a continuación

muestran los canales implementados por PP en esta versión para los dos grupos con un comentario breve.

Canales de mandato

Nombre del canal	***Función***
qmrgload	*Este canal permite iniciar la cola de trabajo cuando el manejador de la cola (QMGR) se carga. Cuando el QMGR esta corriendo entonces utiliza al qmrgload para chequear la consistencia de la cola de trabajo.*
msg-clean	*Después que todo el trabajo ha sido realizado para un determinado mensaje entonces, el QMGR, lo borra de la cola de trabajo usando el canal msg-clean.*
trash	*Este canal es llamado por el QMGR para eliminar todo aquel subdirectorio que este en la cola de trabajo que no se considere de interés.*
timeout	*Cuando un mensaje ha estado en la cola de trabajo más tiempo que el autorizado, el QMGR invoca a este canal para enviarle un mensaje al autor del mensaje original diciéndole cual es el estado del mensaje que ha enviado.*
warning	*El canal warning es usado para enviar reportes cuando el mensaje no se puede enviar por problemas de conexión con el canal.*
drtorfc	*Este canal es usado para construir notificaciones o reportes de errores. Convierte al reporte en el estándar RFC-822 que será enviado al autor.*

Canales opcionales

Nombre del canal	**Función**
822-local	Es utilizado para entregar mensajes en un formato RFC-822 a usuarios locales.
X400-84	Este es un canal de entrada y salida para mensajes con formato X400 P1(84). Corre bajo ISODE.
X400-88	*Este es un canal de entrada y salida para mensajes con formato X400 P1(88). Corre bajo ISODE.*
SMTP	Este canal esta preparado para interactuar sobre el protocolo TCP/IP.
UUCP	Este es un canal para interactuar sobre el protocolo UUCP.
JNTMail	EL JNTMail es usado sobre redes bajo protocolo X25 particularmente UK.
DECNet	Este canal trabaja sobre el protocolo DECNet de Digital.
p2torfc	Sirve para reformatear un encabezamiento de formato X400 P2(84) y P2(88) a RFC-822.
rfctop2	Es la inversa de P2toRFC

fin (la tabla or para direcciones X400 y la tabla domain para direcciones RFC-822). Este proceso de normalización tiene tres posibles salidas. La primera es que la dirección dada no sea reconocida. La segunda es que es reconocida como dirección remota y la tercera es que sea reconocida como dirección local. Si la dirección fue reconocida como dirección remota entonces la información de ruteo es utilizada para acceder a dicha máquina via la s de canal. Si la dirección es reconocida como local entonces esta será buscada en la tabla de alias y si es encontrada se reprocesará esa dirección local accediendo nuevamente a la tabla de alias. Si no fue encontrada esa dirección en la tablas de alias entonces esta es buscada en la tabla de usuarios.

Una vez que tenemos normalizada la dirección y accedimos a la información de ruteo las direcciones son convertidas o transformadas (si es necesario, por ejemplo: de formato X400 a RFC-822) acorde a la especificación RFC-1148, usando las tablas ortorfc, rfctoor y rfc1148gate.

Una vez pasado por todos los puntos anteriores la dirección es válida, lo que faltaría es controlar la autorización que tiene esta dirección de entrar o en su defecto de salir. PP cuanta con información de autorización que será abordada en los próximos párrafos.

La tabla de Alias

La sintaxis general es *clave: tipo valor interpretación* al igual que Sendmail y MMDF. Como ya hemos comentado el manejo de alias nos permite mapear direcciones que no son usuarios a direcciones de usuarios (por ejemplo: postmaster a root). Redireccionar a usuarios que se han trasladado a algún otro lugar. Reescribir las direcciones de usuarios (por eje.:A.G@fisilp.unlp.edu.ar a goeta@fisilp.unlp.edu.ar). Veamos algunos ejemplos:

postmaster: alias root@ayelen.fisica.unlp.edu.ar 822 external
A.G.Russo:synonym arusso 822
A.Russo: alias A.G.Russo 822
G.Punte: alias "/I = Graciela/S = Punte/O = Profimo/PRMD = Dto.Fisica/ADMD = /C = ar/" X400

El tipo synonym nos esta indicando que el nombre de la izquierda será reemplazado por el nuevo valor de la derecha. La diferencia con el tipo alias es que el nombre no será expandido en el mensaje mientras que el synonym sí. La interpretación X400 y 822 hace referencia a los formatos de direcciones X400 y RFC-822 respectivamente. El external nos indica que no se consultará más la tabla de alias y que se continuará con el procesamiento de la dirección.

La tabla de usuarios

Esta tabla nos determina el canal a usar para un usuario local. Su sintaxis es la siguiente: *nombre_usuarios : canal mta, canal mta,...* El canal determina el nombre del canal a utilizar sobre el mta. El mta es el nombre de la máquina local donde el mailbox reside. Veamos el siguiente ejemplo:

arusso:smtp smtp_relay
arusso:822-local ayelen.fisica.unlp.edu.ar

list	Este canal es utilizado para expandir las listas de distribución a todos los miembros que la componen. El mensaje que pasa por este canal es enviado a cada uno de sus miembros.
p2explode	En este canal se fracciona un mensaje en formato P2.
p2flatten	Recompone las fracciones de un mensaje P2.
dirlist	Este canal expande una lista de distribución en los miembros que la componen. Toma a esos miembros del directorio X500.
fcontrol	Es usado para correr varios filtros para los cuerpos de los mensajes. Estos filtros pueden afectar solo al cuerpo y no a todo el mensaje.
splitter	El canal splitter es usado algunas veces para dividir el mensaje en mensajes mas chicos. Esto es usado para los mensajes de FAX.

A continuación daremos dos ejemplos de canales, uno especificará un canal local RFC-822 y el otro un filtro de formato fax a texto.

chan 822-local prog = local,
show = "Local 822 Delivery channel", type = both,
sort = "user time", adr = 822, hdrout = 822, bpout = ia5, content-out = 822,
outtable = local, acces = mts, drchan = dr2rfc
chan g3faxtotxt prog = fcontrol, type = shaper,
bpin = g3fax, bpout = ia5, outinfo = f-fax2txt, show "Fax -< Text mapper"

El parámetro *prog* = determina el programa a ejecutarse, en el primero se ejecutará el programa local y en el segundo fcontrol. El parámetro type = indica si el canal es de salida, de entrada, ambos, de warning, de timeout,..., en nuestro ejemplo el primero es de entrada y salida (both), el segundo esta definido como formateador (shaper). El parámetro ad = determina el formato de dirección de entrada y salida. El parámetro hdrout determina el formato del encabezamiento, en este caso es RFC-822. El parámetro bpout se determina el formato del cuerpo del mensaje de salida, en el primero es RFC-822 y en el segundo es ia5. El parámetro outtable determina en que tabla ha de resolver la dirección local y acces indica la manera de acceso. En el segundo ejemplo el bpin indica el formato de entrada, en este ejemplo es del tipo g3faxtotxt. El parámetro show cumple la misma función que el del MMDF. El parámetro contet-out determina el permiso de mensajes de salida en un determinado formato, en nuestro ejemplo es RFC-822. El parámetro out-info determina un programa e ejecutarse cuando un mensaje sea procesado por este canal. Las combinaciones son variadas, en particular uno puede definir un canal local para entrada y otro para la salida.

Manejo de tablas

Cuando un mensaje arriba, el submit usa las tablas de dominio, de alias, de usuario y de nombres or para identificar las rutas de las direcciones especificadas en el mensaje recibido.

Una dirección arriba en un formato de dirección estándar como RFC-822 o X400. Como primer paso la dirección es normalizada via tablas definidas para tal

profimo_list:list

Este ejemplo muestra que los mensajes que le lleguen a arusso serán entregados por el canal smtp a través de unlp.unlp.edu.ar y por el canal 822-local a través de ayelen.fisica.unlp.edu.ar. Si llegase un mensaje a profimo_list este será procesado por el canal list.

Tabla de dominio (domain)

La tabla de dominios es utilizada por el submit para obtener el dominio completo de cualquier alias o nombres reducidos y para obtener un puntero con información de ruteo. Su sintaxis es la siguiente:

LHS : identificador_de_entrada | identificador_de_entrada
identifacador_de_entrada:: = key [=valor] .....

Para la construcción del LHS (left hand side), existen dos alternativas, la primera que machee exactamente con el valor de entrada y la segunda que machee en forma parcial, utilizando caracteres wildcard.

Explicaremos aquí algunos de los posibles valores de *key.*

valor	*Significado*
norm=nom-dom	*Identifica el dominio completo. Si nom-dom no estuviera presente el nombre del dominio se toma del lhs.*
mta=ref-can	*Provee un puntero para encontrar la información de ruteo para este dominio en las tabla de los canales.*
synonym=nom-dom	*El lhs es reemplazado por nom-dom.*
local=nom-subtbl	*Indica que el lhs es local. Si el nom-subtbl esta especificado este sirve para administrar varios subdominios*

Veamos el siguiente ejemplo:

ayelen.fisica.unlp.edu.ar:local
**.fisica.unlp.edu.ar: mta=smtp_directo*
**.unlp.edu.ar:mta=unlp*
**:mta=smtp_relay*

En este ejemplo estamos indicando que toda aquella dirección que sea usuario@ayelen.fisica.unlp.edu.ar será procesada localmente. Toda dirección como athos.fisica.unlp.edu.ar macheará com *.fisica.unlp.edu.ar con lo que se tratará de buscar el canal a través del puntero al canal smtp_directo . Observe si el dominio dado no machea con ninguno de los tres primeros, seguro que machea con el último que actúa como mail_relay.

Tabla or

Esta tabla provee información para el ruteo e información de las direcciones X400 de la misma forma que la tabla de dominios. Su sintaxis es *key : type value* . La tabla or sirve para evaluar direcciones O/R. El key es un nombre de dominio or según la especificación del RFC-1148. En análisis de búsqueda es a través de un árbol, en caso de que todos los niveles del árbol fallara automáticamente queda descartada la opción. La búsqueda continuará hasta encontrar la mejor opción. Se permite el uso de caracteres wilcard. Veamos el siguiente ejemplo: supongamos que la dirección que consultamos es */S=pac/O=XTel/PRMD=X-Tel Services/ADMD= /C=GB/* y la entrada en la tabla fuera *ADMD$*.C$GB: mta admd.gb*, observe que C$GB machea, y que ADMD$*.C$GB machea, los otros campos fallan con lo cual es seleccionado el mta admd.gb. En caso de haber dos entradas en la tabla que machean parcialmente se selcciona el mta que mejor machea.

Tabla de canales

El submit utiliza este tabla para investigar los posibles canales, estos pueden ser usados para enviar mensajes a una máquina remota. Esta máquina remota esta identificada por una entrada mta en las tablas or o domain. Su sintaxis es la siguiente *key: nombre_del_mta (canal_a_utilizar)* Continuando con los ejemplos vistos en las tablas domain y or, una posible tabla de canal podría ser la siguiente:

smtp_directo:
unlp:unlp.unlp.edu.ar (smtp)
smtp_relay:unlp.unlp.edu.ar (smtp)

Supongamos que queremos enviar un mensaje a arusso@unlp.unlp.edu.ar, PP identifica la dirección como remota entonces tratará de buscar en la taba de dominios el dominio *unlp.unlp.edu.ar* el cual machea con **.unlp.edu.ar*, el valor que contiene es unlp que será buscado en la tabla de canales para ubicar al mta con el cual se conectará para enviar el mensaje, En este caso se conectará via el canal *smtp* a *unlp.unlp.edu.ar*.

Tablas de conversión de direcciones O/R a RFC-822

La tabla ortorfc mapea direcciones O/R en direcciones en formato RFC-822. Veamos el siguiente ejemplo.

PRMD$UK\.AQC.ADMD$GOLD 400.C$GB:ac.uk
ADMD$GOLD 400.C$GB:gold-400.gb

Tablas de conversión de direcciones RFC-822 a O/R

Es la inversa de la anterior. Veamos un ejemplo:

ac.uk:*PRMD$UK\.AQC.ADMD$GOLD 400.C$GB*
gold-400.gb:ADMD$GOLD 400.C$GB

Tablas de entrada y salida para X400

Tabla de salida

Esta tabla es utilizada para los canales de salida sobre X400, permitiéndole mapear la máquina con la información necesaria para el direccionamiento y la conexión con un MTA remoto. La sintaxis es la siguiente: *mta_key: rmta=xx, rpass=xx, rpsap=xx,...* . El campo mta_key representa el nombre del mta que deriva de la tabla de canales. Un ejemplo podría ser el siguiente:

france-reunir:rmta = "sunir.reunir.fr", rpass = " ", rpsap = 'Int-X25(80) = 20803500034996'

En los siguientes párrafos veremos que significado tienen cada uno de los parámetros.

Uso de la tabla de salida

PP se conecta con la máquina remota usando la dirección especificada en el campo rpsap. Cuando se logra la conexión, PP le da al MTA remoto el nombre y la password de identificación. La máquina remota acepta la conexión y retorna su propio nombre y password, o cancela la conexión con un aviso del motivo. Si la conexión es cancelada se aborta todo el proceso, sino, PP machea el nombre del MTA remoto y su password con los valores que contiene la tabla de salida de direcciones X400. Si el nombre del MTA y su password machea, entonces se envia el mensaje, sino se cancela la operación.

Tabla de entrada

Esta tabla es usada por los canales de entrada de X400 para permitirle obtener información del nombre del MTA y información adicional. La sintaxis es similar a la tabla anterior sólo que el campo key contendrá el valor de conexión. Un posible ejemplo podría ser:

"3459"/Int-X25(80) = 208006030203:rname = "france-kwai", rmta = "kwai.inria.fr"

Uso de la tabla de entrada

El MTA remoto pide la conexión con el MTA local y dentro del pedido de conexión se encuentra la dirección de red, el nombre del MTA y la password de acceso. EL PP local usa la dirección de red remota como clave de acceso a la tabla, si la clave es inválida, entonces el PP envia un mensaje de falla y corta la conexión. Si la clave es válida entonces el nombre y la password del MTA remoto es chequeado con los valores que se tienen en la tabla de entrada. Si los valores machean entonces se envian la password y el nombre del MTA local y espera la confirmación del MTA remoto. De confirmar positivamente el MTA remoto, comienza le transferencia sino, se corta la conexión.

Seguridad

La seguridad de PP se puede clasificar en dos aspectos, la primera es la misma que la de Sendmail y MMDF, es decir la seguridad que ofrece el sistema de no contener agujeros negros que pudiesen ser la puerta de entrada a la violación del sistema y la segunda, novedosa, es la de tener niveles de autorización para el uso del MTA local.

En la primera, el PP se puede configurar para que corra bajo un nombre de usuario que sea diferente a root al igual que el MMDF. Esta ventaja permite que si algún intruso lograse violar al sistema, los daños estarían concentrados únicamente en el sistema de correo.

En la segundo aspecto, PP ofrece tres niveles de autorización de entradas la primera a nivel el uso de los canales, la segunda de los MTA's que intervienen y la tercera a nivel de usuarios que intervienen en la transferencia del los mensajes. Las autorizaciones al igual que las otras características que posee PP se encuentran especificadas en tablas, estas son llamadas de autorización que serán comentadas a continuación.

Tabla de autorización de canales

En toda transferencia de mensajes existen dos canales intervinientes, uno de entrada y otro de salida. La tabla de autorización de canales contiene pares de canales (uno de entrada y otro de salida) con parámetros que determinan el control sobre estos. Los parámetros comunes en estas tablas de autorización son las que a continuación expondremos:

Parámetro	**Significado**
none	El mensaje no ha de ser transferido.
free	No existen restricciones.
block	Almenos uno de los cuatro valores debe habilitar el mensaje con le especificación de in o both para la entrada o out o both para la salida. Si ninguno de ellos aparece, el mensaje no ha de ser transferido.
negative	Este implica que pasará todo excepto casos especiales que serán especificados explícitamente.

En particular en la tabla de canales existen otros parámetros tales como:

Parámetro	**Significado**
warnsender = file	Envia un mensaje de warning (definido en file) al autor si por alguna razón de autorización fallase la entrega o procesamiento.
warnrecipient = file	Idem al anterior pero al destinatario.
sizelimit = numero	Mensajes cuyos tamaños excedan la longitud máxima permitida no serán transmitidos.
test	Se permite el paso de los mensajes aún violando el sistema de seguridad y control, pero serán enviados los mensajes de warning en caso de alguna violación a lo definido.

Un ejemplo podría ser el siguiente:

822-local -> x400out84:block, warnsender = error

smtp->smtp:block, sizelimit = 100000
x400in84->x400out84:none

Tablas de autorización de MTA's

Anteriormente dijimos que cuando un mensaje es procesado intervienen dos canales, uno de entrada y otro de salida, pero sobre estos canales existen dos MTA's que actúan tanto en la entrada como en la salida. Esta tabla controla el permiso de determinados MTA's especificados en estas tablas. Existen al igual que en las otras tablas parámetros que son los siguientes:

Parámetro	**Significado**
canal = dirección	Determina la dirección permitida para el MTA dado (in de entrada, out de salida both ambos y none ninguno).
default = dirección	En caso de no estar especificado explícitamente la dirección del canal se utiliza este parámetro por omisión.
sizelimit = numero	Determina la capacidad de envío máximo para este MTA.
bodypart-exclude = valor	Determina cuales tipos de cuerpos de mensajes no estan autorizados a pasar por este MTA.
requieres = pattern	Este MTA exige que la dirección del destinatario y emisor machee con el pattern dado.
exclude = pattern	El MTA requiere que el usuario no machee con el pattern dado.

Vamos un ejemplo:

unlp.unlp.edu.ar:default = both, smtp = out, exclude = "arusso. "*
localhost:default = in, smtp = both,sizelimit = 100000

Tablas de usuarios

En la tabla de usuarios es donde se determinan los derechos de accesos a las direcciones del emisor como la del receptor. Los parámetros que pueden utilizarse son los siguientes:

Parámetro	**Significado**
canal = dirección	Determina la dirección permitida para un usuario dado (in de entrada, out de salida both ambos y none ninguno).
default = dirección	En caso de no estar especificado explícitamente la dirección del canal se utiliza este parámetro por omisión.
sizelimit = numero	Determina la capacidad de envío máximo para este usuario.
bodypart-exclude = valor	Determina cuales tipos de cuerpos de mensajes no estan autorizados a pasar para ese usuario.

Veamos un ejemplo:
P.Quiroga@ayelen.fisica.unlp.edu.ar:default = both
/I = J/S = Taylor/OU = cs/O = ucl/PRMD = uk.ac/ADMD = gold 400/C = gb/:bodypart-exclude = g3fax, sizelimit = 100000

Uso de las tablas de autorización

En primer lugar, se determina cuales son los canales intervinientes (el de entrada y el de salida), estos canales son buscados en las tablas de autorización de canales, si no son encontrados se utiliza el valor por omisión que se encuentra en el archivo de configuración. Si el par de canales tiene el parámetro free, entonces se deja pasar el mensaje. Si el acceso es negativo, se chequea el MTA al igual que las direcciones intervinientes para verificar si explícitamente esa ruta esta desabilitada en esas tablas. Si los canales poseen el parámetro block, se verifica que el MTA y las direcciones no estén explícitamente desabilitadas. Una vez realizado el control de autorización el mensaje puede ingresar o salir de PP.

Listas de usuarios

Las listas de usuarios contienen los mismos conceptos que las del MMDF, la única variante esta en la forma de especificación y su filosofía conceptual. Esta tabla es utilizada por el canal list para expandir un mensaje a sus integrantes. Su sintaxis es la siguiente: *nombre_de_la_lista: [administrador | administrador ....], file = integrantes | dirección, dirección, descripción*. Mientras que en el MMDF se debía definir la lista con el canal de procesamiento de lista, aqui ese enlace esta dado implícitamente. El administrador o administradores se define explícitamente y al igual que el uso del include existe el parámetro file que cumple la misma función que este. Veamos el siguiente ejemplo:

profimo:root@ayelen.fisica.unlp.edu.ar|arusso@unlp.unlp.edu.ar,file = /users/pp/lista,Integrantes del PROFIMO

El PP cuenta con un programa adicional para el control y administración de las listas de usuarios.

Manejo de errores

Existe al igual que en el Sendmail y MMDF un canal *drtorfc* que es el encargado de enviar cualquier reporte de error, pero adicionalmente posee un canal de *warning* que periódicamente le avisará al emisor el estado de su mensaje. Supongamos que un usuario envia un mensaje y este tiene a lo sumo 72hs para ser enviado sino un reporte del error es redireccionado al emisor. El PP puede ser configurado para que en lapsos de 24hs informe (en caso de que el mensaje siga encolado) al emisor del estado del mensaje. El período de mensajes de aviso esta definido en el archivo de configuración.

La cola de trabajo (queue)

PP utiliza también una cola de trabajo para almacenar los mensajes para su posterior procesamiento, con la diferencia que siempre escribe el mensaje en la cola de trabajo. PP tiene un fiel reflejo de la cola de trabajo en la memoria de la

máquina, asegurándole mayor eficiencia en la administración al no tener que realizar búsquedas en disco (controla periódicamente la consistencia).

Una novedad incorporada por PP es la posibilidad limitar la cantidad de mensajes que pueden existir en la cola de trabajo como así también el poder tener más de una cola de trabajo. La estructura de la cola de trabajo difiere del Sendmail y MMDF, al tener un subdirectorio con los mensajes, donde cada mensaje esta representado a su vez por un subdirectorio que contiene subdirectorios con los diferentes alteraciones que el mensaje ha ido sufriendo (conversiones de direcciones, transformaciones del cuerpo del mensaje,...). La cola de trabajo puede manejarse y/o controlarse en forma remota con el uso de ROS y un programa adicional que viene incorporado. Esta ventaja permite que un administrador no necesite logonearse a la máquina para el controlar la cola.

Capacidades adicionales

Capacidad del uso de G3Fax

PP define una serie de canales como filtros entre el formato RFC-822 y el servicio de fax, tanto para el direccionamiento como para el cuerpo del mensaje. Esos son los siguientes *hdr2fax* y *ia5tofax*. El primero procesa la dirección del formato RFC-822 y crea el encabezamiento del fax mientras que el segundo transforma el cuerpo de formato ia5 (texto ASCII) al formato fax. Existen tablas para el control y manejo de este servicio.

Uso de ASN.1

Esta capacidad adicional le permite dividir cuerpos de mensajes en formato ASN.1, ejecutar una serie de conversiones en los cuerpos de los mensajes y procesar una serie de mensajes en formato ASN.1. Existen filtros de transformación de un formato a otro como lo es de ia5 a MOTIS-86-6937 o de teletex a generaltext,....

Archivo de configuración

El archivo de configuración que estamos usando actualmente en la HP-UX 9000/710 es el siguiente:

```
############################################################
#
# Tailor file for PP
#
#
############################################################

############################################################
#
# The following variables will need tailoring
#
############################################################
#loc_dom_mta  mylocalhost.localdomain.us
loc_dom_mta ayelen.fisica.unlp.edu.ar
#
loc_dom_site  ayelen.fisica.unlp.edu.ar
```

```
#
postmaster  root@ayelen.fisica.unlp.edu.ar
#
pplogin   pp
#
qmgrhost  ayelen.fisica.unlp.edu.ar
#
#
loc_or    "/ou=research/o=La Plata University/prmd=Internet/admd= /c=AR/"
delim1    "\1\1\1\1\12"
delim2    "\1\1\1\1\12"
authchannel     block
authloglevel    high
wrndfldir warnings
submit_addr ayelen.fisica.unlp.edu.ar:pp-submit
lockstyle   fcntl
# Body types
headertype  822 822-us
bodypart  ia5
bodypart    binary
quedir /usr/spool/pp
#  **-- Mapping tables --**
#
# All of these table MUST have the names given here
tbl aliases   show="Aliases: mapping -> local id",
tbl users   show="Users: mapping local id -> disposition",
tbl domain    show="Mapping domain key ->full domain/MTA",
tbl or    show="Mapping O/R Address -> MTA",flags=dbm
tbl channel   show="Binding MTA -> Channels",flags=dbm
tbl or2rfc    show="RFC 987: X.400 -> RFC 822",flags=dbm
tbl rfc2or    show="RFC 987: RFC 822 -> X.400",flags=dbm
tbl rfc1148gate show="Gateways that perform RFC 1148 conversions",
tbl auth.channel  show="Authorisation: channel policy",flags=dbm
tbl auth.mta  show="Authorisation: mta based",flags=dbm
tbl auth.user show="Authorisation: user based",flags=dbm
tbl auth.qmgr show="Authorisation: qmgr control",flags=linear
#  **-- Channel tables --**
#
# Needed by individual channels, not submit
tbl local   file="ch.local",
                    show="local id -> user id + home directory",
                    flags=dbm
tbl shell   file="ch.shell", show="info for shell chan"
                    flags=dbm
tbl list    file="ch.list",
                    show="Lists: distribution lists",
                    flags=dbm
tbl uucp    file="ch.uucp" show="UUCP: rfc822 -> uucp", flags=dbm

chan  822-local    prog=local,
    show="Local Delivery channel",type=both,
```

```
    sort = "user time",adr = 822,adr-order = usapref,
    hdrin = 822,hdrout = 822-us, bptout = ia5,
    content-out = 822,outtable = local,access = mts,drchan = dr2rfc

#chan smtp-in   key = "smtp",prog = smtp,show = "with SMTP (PP) IN",type = in
#     adr = 822,hdrin = 822, bptin = ia5
#
#chan smtp-out  key = "smtp",prog = smtp,show = "with SMTP (PP) OUT",type = out,
#     adr = 822,hdrout = 822-us, bptout = ia5,
#     content-out = 822,drchan = dr2rfc

chan  smtp  prog = smtp,show = "with SMTP (PP)",type = both,
            adr = 822,hdrout = 822-us, bptout = ia5,
            content-out = 822,drchan = dr2rfc


chan  list  prog = list,show = "List channel",type = both,
            outtable = list,drchan = dr2rfc,out-info = linked,sort = "user time"

chan  uucp  prog = uucp-out,key = "uucp-out,rmail",
            show = "UUCP outbound channel",type = both,
            adr = 822,adr-order = usapref,outtable = uucp
            out-info = "uux = /tmp/uux,host = xtel"

chan  dr2rfc  prog = dr2rfc,show = "Dr2rfc channel",type = both,
            adr = 822,adr-order = usapref,
            bptin = ia5,hdrin = 822
            content-out = 822,out-info = "return = all"

chan  shell prog = shell,type = out,access = mts
            show = "Shell channel",
            outtable = shell, hdrout = 822-us
            bptout = "ia5", content-out = 822

# QMGR special channels (load/manage/clean)

chan  qmgr-load prog = qmgr-load,show = "Loading the QMGR",
                  type = qmgrload

chan  msg-clean prog = msg-clean,show = "Removing finished message"
                  type = delete

chan  trash   prog = trash,show = "Removing trash"
                  type = debris,out-info = 5h

chan  timeout   prog = timeout,show = "Timeout messages"
                  type = timeout

chan    warning         prog = warnings,show = "Send warning messages",

type = warn
```

```
chan    splitter        prog=splitter,type=split,

show="Divide message into single"


#  **-- shaper channels --**

#  Flatteners

chan  822flatten  prog=rfc934,type=shaper,content-out=822

#  Unflatteners


# Header Filters
chan  822tous   prog=fcontrol,type=shaper,hdrin=822,
                    hdrout=822-us,show="822 -> us",
                    out-info="rfc822norm -822 -littleend"

chan  bin2ia5   prog=fcontrol,type=shaper,bptin=binary,
                    bptout=ia5,outinfo="/usr/bin/uuencode binary",
                    show="Binary -> Text"
#  **-- Logging for the rest of the prog --**
#
authlog level=notice, size=-1
operlog level=notice, size=400
normlog level=notice, size=400, sflags=zero

822flatten  normlog  file=filter
822tous   normlog  file=filter
bin2ia5   normlog  file=filter
dr2rfc    normlog  file=filter
fcontrol  normlog  file=filter
rfc822norm  normlog  file=filter
trash   normlog  file=filter

list    normlog  file=lists

qmgr    normlog  file=qmgr
qmgr-load normlog  file=qmgr

smtpsrvr  normlog  file=smtp
smtp    normlog  file=smtp, level=trace
submit    normlog  file=submit, level=trace

local   normlog  file=local
msg-clean normlog  file=msg-clean

pptsapd   normlog  file=pptsapd,   dlevel=notice
```

Capítulo III
Conclusiones y Cuadro Comparativo

Cuadro Comparativo

	Sendmail	MMDF	PP
Selección del transporte	La selección del transporte la realiza mediante el uso de las reglas. El algoritmo de selección del transporte puede ser alterado por el administrador del MTA.	Procesa la dirección y luego realiza un búsqueda sobre los canales definidos para verificar si este se encuentra en alguno de ellos, en caso de fallar utiliza el canal badhost.	Analiza las tablas de dominios y en caso de machear con alguno de ellos se utiliza el puntero que sirve como entrada a la tabla de canales.
Alias	Son manejados en un archivo por omisión cuyo subdirectorio está especificado en el sendmail.cf. Posee mecanismos ágiles de consulta mediante la indexación de los alias	Los archivos de alias estan definidos en las palabras claves MTBL y ALIAS. Se marca la diferencia entre alias de administración, alias de listas y los alias de usuarios. Permite indexar los alias para una mayor eficiencia en la resolución	El archivo de alias es único y su ubicación esta especificada en el archivo de configuración.
Listas	Introduce implícitamente el concepto de administrador de listas de correo con el uso de la palabra clave include. Por su implementación no permite que el administrador no sea local. No tiene programas de administración de listas.	Introduce explícitamente el concepto de administrador de lista de correo. Posee además la característica de la palabra clave include. Permite que el administrador sea remoto. No tiene programas de administración de listas.	Introduce explícitamente el concepto de administrador de listas. Posee un parámetro o palabra clave file que cumple la misma función que el include. Permite mas de un administrador de la lista que pueden ser adminitradores remotos. Posee un programa para su administración
Cola de trabajo	Cola de trabajo única. No posee capacidad de control sobre la cantidad de mensajes en cola.	Tantas colas de trabajo como canales exista. No posee capacidad de control de mensajes en cola.	Esta subdividida en un subdirectorio por mensaje que a su vez contiene subdirectorios para los archivos frutos de transformaciones. Tiene la cola en memoria. Puede subdividirse la cola de trabajo en varios subniveles. Permite el control de la capacidad de la cola de trabajo.
Control de volumen de información en los transportes	Puede poner límites de volumen de información sobre los transportes. Se especifica en la definición del transporte	No tiene manera de controlarlo	Posee la capacidad no sólo de controlar el volumen de información del transporte sino que controla el volumen de información de los canales, MTA's y usuarios intervinientes.

Seguridad	Alto nivel de seguridad en la última versión que fue obtenido por su evolución. Dispone mecanismos de control sobre el programa principal (sendmail).	Alto nivel de seguridad. Dispone mecanismos de control de acceso a los diferentes programas que son el corazón del MMDF	Sofisticado nivel de seguridad y mecanismos de control a los diferentes programas principales. Posee el control, administración y autorización de los canales, MTA's y usuarios.
Versatilidad con direcciones X400	Permite el parceado de direcciones X400 y su posterior de envío sobre el transporte X400.	No existe mecanismo para el control sobre direcciones X400.	Gran capacidad de control de direcciones X400. Fue específicamente diseñado para su manejo.
Orientación de diseño sobre las direcciones estándares.	Orientado a mensajes con formato estándar RFC-822.	Orientado a mensajes con formato estándar RFC-822	Orientado a mensajes con formato X400 y RFC-822
RFC-822 -> X400 y X400->RFC-822	No esta pensado para ofrecer el servicio de gateway entre RFC-822 y X400	No esta pensado para ofrecer el servicio de gateway entre RFC-822 y X400.	Ofrece el servicio de gateway entre RFC-822 y X400. Posee tablas específicas de mapeo de direcciones X400 a RFC-822 y de RFC-822 a X400
Lugar de origen	América (EEUU)	América (EEUU)	Europa (Londres)
Prevalecencia de estándares en los lugares de origen	Prevalece el estándar RFC-822	Prevalece el estándar RFC-822	Prevalece el estándar X400
Fecha aproximada de presentación	aprox. 1979	aprox. 1984	aprox. 1989
Soporte de múltiples organizaciones en un solo MTA	No esta diseñado para soportar múltiples organizaciones	No esta diseñado para soportar múltiples organizaciones	Una de sus particularidades es el de soportar la administración de múltiples organizaciones.
Capacidades adicionales	La última versión de Sendmail (V8) posee manejo de fax.	ninguna	Tiene fax, ASN.1 y filtros para su manejo y control.
Manejo de diferentes formatos de cuerpos de mensajes	Maneja sólo formato ASCII	Maneja sólo formato ASCII	Reconoce formato ASCII binary, g3fax, voice,...
Transformaciones de formatos	No tiene.	No tiene	Tiene diferentes formas de pasar de un formato a otro por medio del uso de filtros

Conclusiones

El Sendmail evolucionó en una etapa de transición del correo electrónico. Los cambios de los diferentes protocolos, la caótica situación de los formatos de direcciones llevó a su diseñador y creador a dar las herramientas necesarias para que este MTA sea lo suficientemente versátil y adaptativo a los cambios que se estaban dando. Las definiciones de las reglas (que ya hemos visto) fueron la solución. Permitieron a los administradores configurar al Sendmail en función de sus necesidades y adaptarlos a los diferentes formatos de direcciones que iban surgiendo (por ejemplo:X400). Fue quizás uno de los motivos por el cual este MTA fue muy difundido, además de ser de dominio público. La configuración del Sendmail es un tema en si mismo, no resulta nada trivial el configurarlo sobre todo definir el conjunto de reglas que deberán funcionar sincronisadamente para que cumpla su objetivo (entregar y/o rutear mensajes). Los problemas de direccionamiento en cierta medida han ido paulatinamente suavizándose fundamentalmente por que hoy nadie discute los dos estándares de direcciones (RFC-822 y X400). Esta convergencia a lo estándar y el estudio de las reglas llevó a las empresas como así también a las universidades que lo distribuyen a dar un archivo de configuración por omisión que contempla todas las posibilidades de configuración de las reglas. Las reglas que sirvieron para solucionar problemas direccionamiento en aquellos tiempos hoy son un grave inconveniente para los administradores que no usan la configuración por omisión que ofrece este producto. El Sendmail esta diseñado para el estándar RFC-822, basta recordar que fue creado en los EEUU donde el RFC-822 se impuso como estándar de facto al ser adoptado por Internet. Los formatos de direcciones X400 fueron incorporados en el conjunto de las reglas para soportarlo. Como contrapartida el MMDF adopta una filosofía más simplista que el Sendmail al ser creado y diseñado cuando existía en los formatos de direccionamiento una cierta calma.. El MMDF trata de prestar más atención a los transportes intervinientes en la transmisión de los mensajes. Pues si bien el problema de las direcciones estaba casi solucionado no lo era así el del transporte. Lo que el MMDF trató de hacer fue en alguna medida independizarse de la configuración de los transportes con la incorporación del concepto de canales. Este concepto fue generalizandose en el MMDF, incorporandose en él funciones propias de administración. El PP tuvo su presentación en los papeles en el año 1989, adoptando los conceptos volcados por el MMDF, incorpora en su diseño el concepto de canal. Su implementación sobre las capas de ISO le permiten asimilar beneficios ofrecidos por ISO tales como ASN.1 y permitirla interacción con X400. El sistema de correo X400 permite que usuarios envíen diferentes tipos de mensajes de diferentes formatos, es ahí donde PP adopta esos formatos en los llamados canales permitiéndole a su vez el traspaso de un formato a otro. PP no se aísla con X400 sino que adopta también el formato RFC-822 y permite que este sea usado como gateway entre ambos formatos. Este MTA esta siendo usado por las universidades de Europa y comienza a tener repercusión en los EEUU. Este país esta tratando de migrar en forma lenta y progresiva al formato X400 haciendo uso de PP. **Hoy tenemos al PP funcionando en la República Argentina en la UNLP.**

Bibliografía

Electronic Mail Adressing in Theory and Practice
Lennart Lövstrand - Linköping University - Sweden
PP-Manuals
Steve Kille y Julian Onions - X-Tel Services Ltd. and University Collage London
RFC-822
David H. Crocker - NIC
RFC-1028
O. Jacobsen y D.Lynch-NIC
RFC--1506
J. Houttuin - NIC
Sendmail
Bryan Costales, Eric Allman y Neil Rickert - O'Reilly & Associates, Inc
Internetworking with TCP/IP Volume I; Principles, Protocols and Architecture
Douglas E.Comer - Prentice-Hall Internationals Editions
Internetworking with TCP/IP Volume II; Desing, Implementation and Internals
Douglas E.Comer and David L. Stevens - Prentice-Hall Internationals Editions
UNIX Networking
Stephen G.Kochan and Patrick H. Wood - Hayden Books
UUCP
Tim O'Reilly and Grace Todino - O'Reilly & Associates, Inc
Practical Unix Security
Simson Garfinkel and Gene Spafford - O'Reilly & Associates, Inc
The Desing of the UNIX Operating System
Maurice J. Bach - Prentice-Hall Internationals Editions
The Directory of Electronic Mail Addressing and Networks
O'Reilly & Associates, Inc - O'Reilly & Associates, Inc
Redes de Ordenadores
Andrew S. Tanenbaum - Prentice-Hall Internationals Editions
System Administrator Guide
S.C.O Open Desktop - S.C.O Open Desktop
Installing and Administering ARPA Services
Hewlett Packard - HP9000 computers - Hewlett Packard - HP9000 computers
Unix Systema V versión 4
Stephen Coffin - Osborne McGraw-Hill
Understanding and Using Computer Networks
Uday Pabrai, Cary Dowat and Janet Carlson - Fermi National Accelerator Laboratory - Batavia, Illinois.
IRIS 4 - ISODE, una realizaión específica de OSI
Celestino Tomás Guirao - Fundesco
Red IRIS 19-20-21-22
Fundesco
RFC-882
P. Mockapetris - NIC

Nota: Toda la bibliografía esta disponible al igual que los fuentes de PP e ISODE. Las librerías de ISODE estan compiladas en la HP-UX. Cualquier persona que quiera acceder a ellas deberá mandarme un mail a russo@ayelen.fisica.unlp.edu.ar y toda persona que necesite cualquier tipo de consulta en estos temas, me encontrará a su disposición.

Agradecimientos

Quiero agradecer en primer lugar a mis padres y hermanos (Papá, Mamá, Claudia, Pablo y Sebastián), que hicieron posible que viniera a estudiar a la Ciudad de La Plata y que hoy presente este trabajo de grado.

Quiero agradecerle a mi fiel compañera (mi esposa) y a mi hija por el tiempo que no les he dedicado y por haberme soportado mis estados de ánimos, apoyándome en todo.

Quiero agradecer a Hugo, Paula y Claudia (mis socios) que me alentaron a seguir.

Agradezco a mis compañeros de estudio (Mercedes, Luis, Gabriel, Paula, Hugo, Viyi, Horacio,...) con los cuales sorteamos juntos los diferentes escollos de la facultad.

Quiero también agradecerle a los integrantes del PROFIMO (Dr. Julio Marañon, Dra. Graciela Punte, Dr. Blas Rivero, Dr. Oscar Piro, Dr. Adolfo Fantoni, Técnico Gonzalo Miranda, Lic. Andrés Goeta, Lic. Javier Ellena, Lic. Gustavo Echeverría) que ofrecieron todo su equipamiento para hacer realidad este proyecto.

Un agradecimiento especial a mi Director (Lic. Javier Díaz) que dedicó tiempo y esfuerzo para dirigir este proyecto, y a todas aquellas personas que hicieron posible de alguna manera el que hoy termine este documento como broche de mi carrera de la Licenciatura en Informática.